bomb: file format elf32-littlemips

Disassembly of section .init:

00400758 <\_init>:

400758: 3c1c0002 lui gp,0x2

40075c: 279caa38 addiu gp,gp,-21960

400760: 0399e021 addu gp,gp,t9

400764: 27bdffe0 addiu sp,sp,-32

400768: afbf001c sw ra,28(sp)

40076c: afbc0010 sw gp,16(sp)

400770: 8f82808c lw v0,-32628(gp)

400774: 10400003 beqz v0,400784 <\_init+0x2c>

400778: 8f99808c lw t9,-32628(gp)

40077c: 0320f809 jalr t9

400780: 00000000 nop

400784: 04110001 bal 40078c <\_init+0x34>

400788: 00000000 nop

40078c: 3c1c0042 lui gp,0x42

400790: 279cb190 addiu gp,gp,-20080

400794: 8f998018 lw t9,-32744(gp)

400798: 273908d0 addiu t9,t9,2256

40079c: 0411004c bal 4008d0 <frame\_dummy>

4007a0: 00000000 nop

4007a4: 04110001 bal 4007ac <\_init+0x54>

4007a8: 00000000 nop

4007ac: 3c1c0042 lui gp,0x42

4007b0: 279cb190 addiu gp,gp,-20080

4007b4: 8f998018 lw t9,-32744(gp)

4007b8: 27392440 addiu t9,t9,9280

4007bc: 04110720 bal 402440 <\_\_do\_global\_ctors\_aux>

4007c0: 00000000 nop

4007c4: 8fbf001c lw ra,28(sp)

4007c8: 03e00008 jr ra

4007cc: 27bd0020 addiu sp,sp,32

Disassembly of section .text:

004007d0 <\_\_start>:

4007d0: 03e00021 move zero,ra

4007d4: 04110001 bal 4007dc <\_\_start+0xc>

4007d8: 00000000 nop

4007dc: 3c1c0002 lui gp,0x2

4007e0: 279ca9b4 addiu gp,gp,-22092

4007e4: 039fe021 addu gp,gp,ra

4007e8: 0000f821 move ra,zero

4007ec: 8f848030 lw a0,-32720(gp)

4007f0: 8fa50000 lw a1,0(sp)

4007f4: 27a60004 addiu a2,sp,4

4007f8: 2401fff8 li at,-8

4007fc: 03a1e824 and sp,sp,at

400800: 27bdffe0 addiu sp,sp,-32

400804: 8f87805c lw a3,-32676(gp)

400808: 8f888094 lw t0,-32620(gp)

40080c: afa80010 sw t0,16(sp)

400810: afa20014 sw v0,20(sp)

400814: afbd0018 sw sp,24(sp)

400818: 8f998078 lw t9,-32648(gp)

40081c: 0320f809 jalr t9

400820: 00000000 nop

00400824 <hlt>:

400824: 1000ffff b 400824 <hlt>

400828: 00000000 nop

40082c: 00000000 nop

00400830 <\_\_do\_global\_dtors\_aux>:

400830: 27bdffd0 addiu sp,sp,-48

400834: afb30028 sw s3,40(sp)

400838: 3c130041 lui s3,0x41

40083c: 92623230 lbu v0,12848(s3)

400840: afbf002c sw ra,44(sp)

400844: afb20024 sw s2,36(sp)

400848: afb10020 sw s1,32(sp)

40084c: 14400019 bnez v0,4008b4 <\_\_do\_global\_dtors\_aux+0x84>

400850: afb0001c sw s0,28(sp)

400854: 3c120041 lui s2,0x41

400858: 3c110041 lui s1,0x41

40085c: 26523060 addiu s2,s2,12384

400860: 3c100041 lui s0,0x41

400864: 26313064 addiu s1,s1,12388

400868: 02328823 subu s1,s1,s2

40086c: 8e023234 lw v0,12852(s0)

400870: 00118883 sra s1,s1,0x2

400874: 2631ffff addiu s1,s1,-1

400878: 0051182b sltu v1,v0,s1

40087c: 5060000c beqzl v1,4008b0 <\_\_do\_global\_dtors\_aux+0x80>

400880: 24020001 li v0,1

400884: 24420001 addiu v0,v0,1

400888: 00021880 sll v1,v0,0x2

40088c: 02431821 addu v1,s2,v1

400890: 8c790000 lw t9,0(v1)

400894: 0320f809 jalr t9

400898: ae023234 sw v0,12852(s0)

40089c: 8e023234 lw v0,12852(s0)

4008a0: 0051182b sltu v1,v0,s1

4008a4: 1460fff8 bnez v1,400888 <\_\_do\_global\_dtors\_aux+0x58>

4008a8: 24420001 addiu v0,v0,1

4008ac: 24020001 li v0,1

4008b0: a2623230 sb v0,12848(s3)

4008b4: 8fbf002c lw ra,44(sp)

4008b8: 8fb30028 lw s3,40(sp)

4008bc: 8fb20024 lw s2,36(sp)

4008c0: 8fb10020 lw s1,32(sp)

4008c4: 8fb0001c lw s0,28(sp)

4008c8: 03e00008 jr ra

4008cc: 27bd0030 addiu sp,sp,48

004008d0 <frame\_dummy>:

4008d0: 3c040041 lui a0,0x41

4008d4: 8c823068 lw v0,12392(a0)

4008d8: 3c1c0042 lui gp,0x42

4008dc: 10400006 beqz v0,4008f8 <frame\_dummy+0x28>

4008e0: 279cb190 addiu gp,gp,-20080

4008e4: 8f998088 lw t9,-32632(gp)

4008e8: 13200003 beqz t9,4008f8 <frame\_dummy+0x28>

4008ec: 00000000 nop

4008f0: 03200008 jr t9

4008f4: 24843068 addiu a0,a0,12392

4008f8: 03e00008 jr ra

4008fc: 00000000 nop

00400900 <main>:

400900: 27bdffd0 addiu sp,sp,-48

400904: afbf002c sw ra,44(sp)

400908: afbe0028 sw s8,40(sp)

40090c: 03a0f021 move s8,sp

400910: 3c1c0042 lui gp,0x42

400914: 279cb190 addiu gp,gp,-20080

400918: afbc0010 sw gp,16(sp)

40091c: afc40030 sw a0,48(s8)

400920: afc50034 sw a1,52(s8)

400924: 8fc30030 lw v1,48(s8)

400928: 24020001 li v0,1

40092c: 14620009 bne v1,v0,400954 <main+0x54>

400930: 00000000 nop

400934: 8f82804c lw v0,-32692(gp)

400938: 00000000 nop

40093c: 8c430000 lw v1,0(v0)

400940: 8f828050 lw v0,-32688(gp)

400944: 00000000 nop

400948: ac430000 sw v1,0(v0)

40094c: 10000049 b 400a74 <main+0x174>

400950: 00000000 nop

400954: 8fc30030 lw v1,48(s8)

400958: 24020002 li v0,2

40095c: 14620032 bne v1,v0,400a28 <main+0x128>

400960: 00000000 nop

400964: 8fc20034 lw v0,52(s8)

400968: 00000000 nop

40096c: 24420004 addiu v0,v0,4

400970: 8c420000 lw v0,0(v0)

400974: 00000000 nop

400978: 00401821 move v1,v0

40097c: 3c020040 lui v0,0x40

400980: 24422600 addiu v0,v0,9728

400984: 00602021 move a0,v1

400988: 00402821 move a1,v0

40098c: 8f828048 lw v0,-32696(gp)

400990: 00000000 nop

400994: 0040c821 move t9,v0

400998: 0320f809 jalr t9

40099c: 00000000 nop

4009a0: 8fdc0010 lw gp,16(s8)

4009a4: 00401821 move v1,v0

4009a8: 8f828050 lw v0,-32688(gp)

4009ac: 00000000 nop

4009b0: ac430000 sw v1,0(v0)

4009b4: 8f828050 lw v0,-32688(gp)

4009b8: 00000000 nop

4009bc: 8c420000 lw v0,0(v0)

4009c0: 00000000 nop

4009c4: 1440002b bnez v0,400a74 <main+0x174>

4009c8: 00000000 nop

4009cc: 3c020040 lui v0,0x40

4009d0: 24442604 addiu a0,v0,9732

4009d4: 8fc20034 lw v0,52(s8)

4009d8: 00000000 nop

4009dc: 8c430000 lw v1,0(v0)

4009e0: 8fc20034 lw v0,52(s8)

4009e4: 00000000 nop

4009e8: 24420004 addiu v0,v0,4

4009ec: 8c420000 lw v0,0(v0)

4009f0: 00602821 move a1,v1

4009f4: 00403021 move a2,v0

4009f8: 8f828054 lw v0,-32684(gp)

4009fc: 00000000 nop

400a00: 0040c821 move t9,v0

400a04: 0320f809 jalr t9

400a08: 00000000 nop

400a0c: 8fdc0010 lw gp,16(s8)

400a10: 24040008 li a0,8

400a14: 8f828034 lw v0,-32716(gp)

400a18: 00000000 nop

400a1c: 0040c821 move t9,v0

400a20: 0320f809 jalr t9

400a24: 00000000 nop

400a28: 3c020040 lui v0,0x40

400a2c: 24432624 addiu v1,v0,9764

400a30: 8fc20034 lw v0,52(s8)

400a34: 00000000 nop

400a38: 8c420000 lw v0,0(v0)

400a3c: 00602021 move a0,v1

400a40: 00402821 move a1,v0

400a44: 8f828054 lw v0,-32684(gp)

400a48: 00000000 nop

400a4c: 0040c821 move t9,v0

400a50: 0320f809 jalr t9

400a54: 00000000 nop

400a58: 8fdc0010 lw gp,16(s8)

400a5c: 24040008 li a0,8

400a60: 8f828034 lw v0,-32716(gp)

400a64: 00000000 nop

400a68: 0040c821 move t9,v0

400a6c: 0320f809 jalr t9

400a70: 00000000 nop

400a74: 0c10077f jal 401dfc <initialize\_bomb>

400a78: 00000000 nop

400a7c: 8fdc0010 lw gp,16(s8)

400a80: 3c020040 lui v0,0x40

400a84: 24442640 addiu a0,v0,9792

400a88: 8f828038 lw v0,-32712(gp)

400a8c: 00000000 nop

400a90: 0040c821 move t9,v0

400a94: 0320f809 jalr t9

400a98: 00000000 nop

400a9c: 8fdc0010 lw gp,16(s8)

400aa0: 3c020040 lui v0,0x40

400aa4: 24422660 addiu v0,v0,9824

400aa8: 00402021 move a0,v0

400aac: 8f858074 lw a1,-32652(gp)

400ab0: 8f828040 lw v0,-32704(gp)

400ab4: 00000000 nop

400ab8: 0040c821 move t9,v0

400abc: 0320f809 jalr t9

400ac0: 00000000 nop

400ac4: 8fdc0010 lw gp,16(s8)

400ac8: afc0001c sw zero,28(s8)

400acc: 8f848074 lw a0,-32652(gp)

400ad0: 8f828060 lw v0,-32672(gp)

400ad4: 00000000 nop

400ad8: 0040c821 move t9,v0

400adc: 0320f809 jalr t9

400ae0: 00000000 nop

400ae4: 8fdc0010 lw gp,16(s8)

400ae8: 2442ffff addiu v0,v0,-1

400aec: afc20018 sw v0,24(s8)

400af0: 10000018 b 400b54 <main+0x254>

400af4: 00000000 nop

400af8: 2403000b li v1,11

400afc: 8fc2001c lw v0,28(s8)

400b00: 00000000 nop

400b04: 00621023 subu v0,v1,v0

400b08: 8fc30018 lw v1,24(s8)

400b0c: 8f848074 lw a0,-32652(gp)

400b10: 00000000 nop

400b14: 00831821 addu v1,a0,v1

400b18: 80630000 lb v1,0(v1)

400b1c: 00000000 nop

400b20: 2463ffd0 addiu v1,v1,-48

400b24: 8f84806c lw a0,-32660(gp)

400b28: 00021080 sll v0,v0,0x2

400b2c: 00821021 addu v0,a0,v0

400b30: ac430000 sw v1,0(v0)

400b34: 8fc2001c lw v0,28(s8)

400b38: 00000000 nop

400b3c: 24420001 addiu v0,v0,1

400b40: afc2001c sw v0,28(s8)

400b44: 8fc20018 lw v0,24(s8)

400b48: 00000000 nop

400b4c: 2442ffff addiu v0,v0,-1

400b50: afc20018 sw v0,24(s8)

400b54: 8fc2001c lw v0,28(s8)

400b58: 00000000 nop

400b5c: 2842000c slti v0,v0,12

400b60: 1440ffe5 bnez v0,400af8 <main+0x1f8>

400b64: 00000000 nop

400b68: 3c020040 lui v0,0x40

400b6c: 24442664 addiu a0,v0,9828

400b70: 8f828038 lw v0,-32712(gp)

400b74: 00000000 nop

400b78: 0040c821 move t9,v0

400b7c: 0320f809 jalr t9

400b80: 00000000 nop

400b84: 8fdc0010 lw gp,16(s8)

400b88: 3c020040 lui v0,0x40

400b8c: 244426a0 addiu a0,v0,9888

400b90: 8f828038 lw v0,-32712(gp)

400b94: 00000000 nop

400b98: 0040c821 move t9,v0

400b9c: 0320f809 jalr t9

400ba0: 00000000 nop

400ba4: 8fdc0010 lw gp,16(s8)

400ba8: 0c1007fb jal 401fec <read\_line>

400bac: 00000000 nop

400bb0: 8fdc0010 lw gp,16(s8)

400bb4: afc20020 sw v0,32(s8)

400bb8: 8fc40020 lw a0,32(s8)

400bbc: 0c10035b jal 400d6c <phase\_1>

400bc0: 00000000 nop

400bc4: 8fdc0010 lw gp,16(s8)

400bc8: 0c100899 jal 402264 <phase\_defused>

400bcc: 00000000 nop

400bd0: 8fdc0010 lw gp,16(s8)

400bd4: 3c020040 lui v0,0x40

400bd8: 244426cc addiu a0,v0,9932

400bdc: 8f828038 lw v0,-32712(gp)

400be0: 00000000 nop

400be4: 0040c821 move t9,v0

400be8: 0320f809 jalr t9

400bec: 00000000 nop

400bf0: 8fdc0010 lw gp,16(s8)

400bf4: 0c1007fb jal 401fec <read\_line>

400bf8: 00000000 nop

400bfc: 8fdc0010 lw gp,16(s8)

400c00: afc20020 sw v0,32(s8)

400c04: 8fc40020 lw a0,32(s8)

400c08: 0c10036f jal 400dbc <phase\_2>

400c0c: 00000000 nop

400c10: 8fdc0010 lw gp,16(s8)

400c14: 0c100899 jal 402264 <phase\_defused>

400c18: 00000000 nop

400c1c: 8fdc0010 lw gp,16(s8)

400c20: 3c020040 lui v0,0x40

400c24: 244426f8 addiu a0,v0,9976

400c28: 8f828038 lw v0,-32712(gp)

400c2c: 00000000 nop

400c30: 0040c821 move t9,v0

400c34: 0320f809 jalr t9

400c38: 00000000 nop

400c3c: 8fdc0010 lw gp,16(s8)

400c40: 0c1007fb jal 401fec <read\_line>

400c44: 00000000 nop

400c48: 8fdc0010 lw gp,16(s8)

400c4c: afc20020 sw v0,32(s8)

400c50: 8fc40020 lw a0,32(s8)

400c54: 0c1003b5 jal 400ed4 <phase\_3>

400c58: 00000000 nop

400c5c: 8fdc0010 lw gp,16(s8)

400c60: 0c100899 jal 402264 <phase\_defused>

400c64: 00000000 nop

400c68: 8fdc0010 lw gp,16(s8)

400c6c: 3c020040 lui v0,0x40

400c70: 24442718 addiu a0,v0,10008

400c74: 8f828038 lw v0,-32712(gp)

400c78: 00000000 nop

400c7c: 0040c821 move t9,v0

400c80: 0320f809 jalr t9

400c84: 00000000 nop

400c88: 8fdc0010 lw gp,16(s8)

400c8c: 0c1007fb jal 401fec <read\_line>

400c90: 00000000 nop

400c94: 8fdc0010 lw gp,16(s8)

400c98: afc20020 sw v0,32(s8)

400c9c: 8fc40020 lw a0,32(s8)

400ca0: 0c1004af jal 4012bc <phase\_4>

400ca4: 00000000 nop

400ca8: 8fdc0010 lw gp,16(s8)

400cac: 0c100899 jal 402264 <phase\_defused>

400cb0: 00000000 nop

400cb4: 8fdc0010 lw gp,16(s8)

400cb8: 3c020040 lui v0,0x40

400cbc: 24442728 addiu a0,v0,10024

400cc0: 8f828038 lw v0,-32712(gp)

400cc4: 00000000 nop

400cc8: 0040c821 move t9,v0

400ccc: 0320f809 jalr t9

400cd0: 00000000 nop

400cd4: 8fdc0010 lw gp,16(s8)

400cd8: 0c1007fb jal 401fec <read\_line>

400cdc: 00000000 nop

400ce0: 8fdc0010 lw gp,16(s8)

400ce4: afc20020 sw v0,32(s8)

400ce8: 8fc40020 lw a0,32(s8)

400cec: 0c1004fa jal 4013e8 <phase\_5>

400cf0: 00000000 nop

400cf4: 8fdc0010 lw gp,16(s8)

400cf8: 0c100899 jal 402264 <phase\_defused>

400cfc: 00000000 nop

400d00: 8fdc0010 lw gp,16(s8)

400d04: 3c020040 lui v0,0x40

400d08: 2444274c addiu a0,v0,10060

400d0c: 8f828038 lw v0,-32712(gp)

400d10: 00000000 nop

400d14: 0040c821 move t9,v0

400d18: 0320f809 jalr t9

400d1c: 00000000 nop

400d20: 8fdc0010 lw gp,16(s8)

400d24: 0c1007fb jal 401fec <read\_line>

400d28: 00000000 nop

400d2c: 8fdc0010 lw gp,16(s8)

400d30: afc20020 sw v0,32(s8)

400d34: 8fc40020 lw a0,32(s8)

400d38: 0c100540 jal 401500 <phase\_6>

400d3c: 00000000 nop

400d40: 8fdc0010 lw gp,16(s8)

400d44: 0c100899 jal 402264 <phase\_defused>

400d48: 00000000 nop

400d4c: 8fdc0010 lw gp,16(s8)

400d50: 00001021 move v0,zero

400d54: 03c0e821 move sp,s8

400d58: 8fbf002c lw ra,44(sp)

400d5c: 8fbe0028 lw s8,40(sp)

400d60: 27bd0030 addiu sp,sp,48

400d64: 03e00008 jr ra

400d68: 00000000 nop

**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

**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

004018a4 <fun7>:

4018a4: 27bdffe0 addiu sp,sp,-32

4018a8: afbf001c sw ra,28(sp)

4018ac: afbe0018 sw s8,24(sp)

4018b0: 03a0f021 move s8,sp

4018b4: afc40020 sw a0,32(s8)

4018b8: afc50024 sw a1,36(s8)

4018bc: 8fc20020 lw v0,32(s8)

4018c0: 00000000 nop

4018c4: 14400004 bnez v0,4018d8 <fun7+0x34>

4018c8: 00000000 nop

4018cc: 2402ffff li v0,-1

4018d0: 10000029 b 401978 <fun7+0xd4>

4018d4: 00000000 nop

4018d8: 8fc20020 lw v0,32(s8)

4018dc: 00000000 nop

4018e0: 8c430000 lw v1,0(v0)

4018e4: 8fc20024 lw v0,36(s8)

4018e8: 00000000 nop

4018ec: 0043102a slt v0,v0,v1

4018f0: 1040000c beqz v0,401924 <fun7+0x80>

4018f4: 00000000 nop

4018f8: 8fc20020 lw v0,32(s8)

4018fc: 00000000 nop

401900: 8c420004 lw v0,4(v0)

401904: 00000000 nop

401908: 00402021 move a0,v0

40190c: 8fc50024 lw a1,36(s8)

401910: 0c100629 jal 4018a4 <fun7>

401914: 00000000 nop

401918: 00021040 sll v0,v0,0x1

40191c: 10000016 b 401978 <fun7+0xd4>

401920: 00000000 nop

401924: 8fc20020 lw v0,32(s8)

401928: 00000000 nop

40192c: 8c430000 lw v1,0(v0)

401930: 8fc20024 lw v0,36(s8)

401934: 00000000 nop

401938: 0062102a slt v0,v1,v0

40193c: 1040000d beqz v0,401974 <fun7+0xd0>

401940: 00000000 nop

401944: 8fc20020 lw v0,32(s8)

401948: 00000000 nop

40194c: 8c420008 lw v0,8(v0)

401950: 00000000 nop

401954: 00402021 move a0,v0

401958: 8fc50024 lw a1,36(s8)

40195c: 0c100629 jal 4018a4 <fun7>

401960: 00000000 nop

401964: 00021040 sll v0,v0,0x1

401968: 24420001 addiu v0,v0,1

40196c: 10000002 b 401978 <fun7+0xd4>

401970: 00000000 nop

401974: 00001021 move v0,zero

401978: 03c0e821 move sp,s8

40197c: 8fbf001c lw ra,28(sp)

401980: 8fbe0018 lw s8,24(sp)

401984: 27bd0020 addiu sp,sp,32

401988: 03e00008 jr ra

40198c: 00000000 nop

**00401990 <secret\_phase>:**

401990: 27bdffd8 addiu sp,sp,-40

401994: afbf0024 sw ra,36(sp)

401998: afbe0020 sw s8,32(sp)

40199c: 03a0f021 move s8,sp

4019a0: 3c1c0042 lui gp,0x42

4019a4: 279cb190 addiu gp,gp,-20080

4019a8: afbc0010 sw gp,16(sp)

4019ac: 0c1007fb jal 401fec <read\_line>

4019b0: 00000000 nop

4019b4: 8fdc0010 lw gp,16(s8)

4019b8: afc2001c sw v0,28(s8)

4019bc: 8fc2001c lw v0,28(s8)

4019c0: 00000000 nop

4019c4: 00402021 move a0,v0

4019c8: 00002821 move a1,zero

4019cc: 2406000a li a2,10

4019d0: 8f828070 lw v0,-32656(gp)

4019d4: 00000000 nop

4019d8: 0040c821 move t9,v0

4019dc: 0320f809 jalr t9

4019e0: 00000000 nop

4019e4: 8fdc0010 lw gp,16(s8)

4019e8: afc20018 sw v0,24(s8)

4019ec: 8fc20018 lw v0,24(s8)

4019f0: 00000000 nop

4019f4: 2442ffff addiu v0,v0,-1

4019f8: 2c4203e9 sltiu v0,v0,1001

4019fc: 14400004 bnez v0,401a10 <secret\_phase+0x80>

401a00: 00000000 nop

401a04: 0c10087c jal 4021f0 <explode\_bomb>

401a08: 00000000 nop

401a0c: 8fdc0010 lw gp,16(s8)

401a10: 3c020041 lui v0,0x41

401a14: 24443184 addiu a0,v0,12676

401a18: 8fc50018 lw a1,24(s8)

401a1c: 0c100629 jal 4018a4 <fun7>

401a20: 00000000 nop

401a24: 8fdc0010 lw gp,16(s8)

401a28: 00401821 move v1,v0

401a2c: 24020007 li v0,7

401a30: 10620004 beq v1,v0,401a44 <secret\_phase+0xb4>

401a34: 00000000 nop

401a38: 0c10087c jal 4021f0 <explode\_bomb>

401a3c: 00000000 nop

401a40: 8fdc0010 lw gp,16(s8)

401a44: 3c020040 lui v0,0x40

401a48: 244427b8 addiu a0,v0,10168

401a4c: 8f828038 lw v0,-32712(gp)

401a50: 00000000 nop

401a54: 0040c821 move t9,v0

401a58: 0320f809 jalr t9

401a5c: 00000000 nop

401a60: 8fdc0010 lw gp,16(s8)

401a64: 0c100899 jal 402264 <phase\_defused>

401a68: 00000000 nop

401a6c: 8fdc0010 lw gp,16(s8)

401a70: 03c0e821 move sp,s8

401a74: 8fbf0024 lw ra,36(sp)

401a78: 8fbe0020 lw s8,32(sp)

401a7c: 27bd0028 addiu sp,sp,40

401a80: 03e00008 jr ra

401a84: 00000000 nop

00401a88 <sig\_handler>:

401a88: 27bdffe0 addiu sp,sp,-32

401a8c: afbf001c sw ra,28(sp)

401a90: afbe0018 sw s8,24(sp)

401a94: 03a0f021 move s8,sp

401a98: 3c1c0042 lui gp,0x42

401a9c: 279cb190 addiu gp,gp,-20080

401aa0: afbc0010 sw gp,16(sp)

401aa4: 3c020040 lui v0,0x40

401aa8: 244427e0 addiu a0,v0,10208

401aac: 8f828038 lw v0,-32712(gp)

401ab0: 00000000 nop

401ab4: 0040c821 move t9,v0

401ab8: 0320f809 jalr t9

401abc: 00000000 nop

401ac0: 8fdc0010 lw gp,16(s8)

401ac4: 24040003 li a0,3

401ac8: 8f828044 lw v0,-32700(gp)

401acc: 00000000 nop

401ad0: 0040c821 move t9,v0

401ad4: 0320f809 jalr t9

401ad8: 00000000 nop

401adc: 8fdc0010 lw gp,16(s8)

401ae0: 3c020040 lui v0,0x40

401ae4: 24422818 addiu v0,v0,10264

401ae8: 00402021 move a0,v0

401aec: 8f828054 lw v0,-32684(gp)

401af0: 00000000 nop

401af4: 0040c821 move t9,v0

401af8: 0320f809 jalr t9

401afc: 00000000 nop

401b00: 8fdc0010 lw gp,16(s8)

401b04: 00000000 nop

401b08: 8f82803c lw v0,-32708(gp)

401b0c: 00000000 nop

401b10: 8c420000 lw v0,0(v0)

401b14: 00000000 nop

401b18: 00402021 move a0,v0

401b1c: 8f828068 lw v0,-32664(gp)

401b20: 00000000 nop

401b24: 0040c821 move t9,v0

401b28: 0320f809 jalr t9

401b2c: 00000000 nop

401b30: 8fdc0010 lw gp,16(s8)

401b34: 24040001 li a0,1

401b38: 8f828044 lw v0,-32700(gp)

401b3c: 00000000 nop

401b40: 0040c821 move t9,v0

401b44: 0320f809 jalr t9

401b48: 00000000 nop

401b4c: 8fdc0010 lw gp,16(s8)

401b50: 3c020040 lui v0,0x40

401b54: 24442820 addiu a0,v0,10272

401b58: 8f828038 lw v0,-32712(gp)

401b5c: 00000000 nop

401b60: 0040c821 move t9,v0

401b64: 0320f809 jalr t9

401b68: 00000000 nop

401b6c: 8fdc0010 lw gp,16(s8)

401b70: 24040010 li a0,16

401b74: 8f828034 lw v0,-32716(gp)

401b78: 00000000 nop

401b7c: 0040c821 move t9,v0

401b80: 0320f809 jalr t9

401b84: 00000000 nop

00401b88 <invalid\_phase>:

401b88: 27bdfff8 addiu sp,sp,-8

401b8c: afbe0004 sw s8,4(sp)

401b90: 03a0f021 move s8,sp

401b94: 03c0e821 move sp,s8

401b98: 8fbe0004 lw s8,4(sp)

401b9c: 27bd0008 addiu sp,sp,8

401ba0: 03e00008 jr ra

401ba4: 00000000 nop

00401ba8 <read\_six\_numbers>:

401ba8: 27bdffd0 addiu sp,sp,-48

401bac: afbf002c sw ra,44(sp)

401bb0: afbe0028 sw s8,40(sp)

401bb4: 03a0f021 move s8,sp

401bb8: 3c1c0042 lui gp,0x42

401bbc: 279cb190 addiu gp,gp,-20080

401bc0: afbc0020 sw gp,32(sp)

401bc4: afc40030 sw a0,48(s8)

401bc8: afc50034 sw a1,52(s8)

401bcc: 8fc40030 lw a0,48(s8)

401bd0: 3c020040 lui v0,0x40

401bd4: 24432828 addiu v1,v0,10280

401bd8: 8fc20034 lw v0,52(s8)

401bdc: 00000000 nop

401be0: 24420004 addiu v0,v0,4

401be4: 8fc50034 lw a1,52(s8)

401be8: 00000000 nop

401bec: 24a80008 addiu t0,a1,8

401bf0: 8fc50034 lw a1,52(s8)

401bf4: 00000000 nop

401bf8: 24a7000c addiu a3,a1,12

401bfc: 8fc50034 lw a1,52(s8)

401c00: 00000000 nop

401c04: 24a60010 addiu a2,a1,16

401c08: 8fc50034 lw a1,52(s8)

401c0c: 00000000 nop

401c10: 24a50014 addiu a1,a1,20

401c14: afa80010 sw t0,16(sp)

401c18: afa70014 sw a3,20(sp)

401c1c: afa60018 sw a2,24(sp)

401c20: afa5001c sw a1,28(sp)

401c24: 00602821 move a1,v1

401c28: 8fc60034 lw a2,52(s8)

401c2c: 00403821 move a3,v0 //把a3=V0

401c30: 8f828084 lw v0,-32636(gp)

401c34: 00000000 nop

401c38: 0040c821 move t9,v0 //把后面的装到前面t9= v0

401c3c: 0320f809 jalr t9 //此处看a0,输入存入了a0

401c40: 00000000 nop

401c44: 8fdc0020 lw gp,32(s8)

401c48: 28420006 slti v0,v0,6 //V0的数位大于等于6则V0为0，跳转，小于6则V0等于1，不跳转

401c4c: 10400004 beqz v0,401c60 <read\_six\_numbers+0xb8>

401c50: 00000000 nop

401c54: 0c10087c jal 4021f0 <explode\_bomb>

401c58: 00000000 nop

401c5c: 8fdc0020 lw gp,32(s8)

401c60: 03c0e821 move sp,s8

401c64: 8fbf002c lw ra,44(sp)

401c68: 8fbe0028 lw s8,40(sp)

401c6c: 27bd0030 addiu sp,sp,48

401c70: 03e00008 jr ra

401c74: 00000000 nop

00401c78 <string\_length>:

401c78: 27bdffe8 addiu sp,sp,-24

401c7c: afbe0014 sw s8,20(sp)

401c80: 03a0f021 move s8,sp

401c84: afc40018 sw a0,24(s8)

401c88: 8fc20018 lw v0,24(s8)

401c8c: 00000000 nop

401c90: afc2000c sw v0,12(s8)

401c94: afc00008 sw zero,8(s8)

401c98: 10000005 b 401cb0 <string\_length+0x38>

401c9c: 00000000 nop

401ca0: 8fc20008 lw v0,8(s8)

401ca4: 00000000 nop

401ca8: 24420001 addiu v0,v0,1

401cac: afc20008 sw v0,8(s8)

401cb0: 8fc2000c lw v0,12(s8)

401cb4: 00000000 nop

401cb8: 80420000 lb v0,0(v0)

401cbc: 00000000 nop

401cc0: 0002102b sltu v0,zero,v0

401cc4: 304200ff andi v0,v0,0xff

401cc8: 8fc3000c lw v1,12(s8)

401ccc: 00000000 nop

401cd0: 24630001 addiu v1,v1,1

401cd4: afc3000c sw v1,12(s8)

401cd8: 1440fff1 bnez v0,401ca0 <string\_length+0x28>

401cdc: 00000000 nop

401ce0: 8fc20008 lw v0,8(s8)

401ce4: 03c0e821 move sp,s8

401ce8: 8fbe0014 lw s8,20(sp)

401cec: 27bd0018 addiu sp,sp,24

401cf0: 03e00008 jr ra

401cf4: 00000000 nop

00401cf8 <strings\_not\_equal>:

401cf8: 27bdffd0 addiu sp,sp,-48

401cfc: afbf002c sw ra,44(sp)

401d00: afbe0028 sw s8,40(sp)

401d04: 03a0f021 move s8,sp

401d08: afc40030 sw a0,48(s8)

401d0c: afc50034 sw a1,52(s8)

401d10: 8fc40030 lw a0,48(s8)

401d14: 0c10071e jal 401c78 <string\_length>

401d18: 00000000 nop

401d1c: afc20024 sw v0,36(s8)

401d20: 8fc40034 lw a0,52(s8)

401d24: 0c10071e jal 401c78 <string\_length>

401d28: 00000000 nop

401d2c: afc20020 sw v0,32(s8)

401d30: 8fc20030 lw v0,48(s8)

401d34: 00000000 nop

401d38: afc2001c sw v0,28(s8)

401d3c: 8fc20034 lw v0,52(s8)

401d40: 00000000 nop

401d44: afc20018 sw v0,24(s8)

401d48: 8fc30024 lw v1,36(s8)

401d4c: 8fc20020 lw v0,32(s8)

401d50: 00000000 nop

401d54: 1062001b beq v1,v0,401dc4 <strings\_not\_equal+0xcc>

401d58: 00000000 nop

401d5c: 24020001 li v0,1

401d60: 10000020 b 401de4 <strings\_not\_equal+0xec>

401d64: 00000000 nop

401d68: 8fc2001c lw v0,28(s8)

401d6c: 00000000 nop

401d70: 80430000 lb v1,0(v0)

401d74: 8fc20018 lw v0,24(s8)

401d78: 00000000 nop

401d7c: 80420000 lb v0,0(v0)

401d80: 00000000 nop

401d84: 00621026 xor v0,v1,v0

401d88: 0002102b sltu v0,zero,v0

401d8c: 304200ff andi v0,v0,0xff

401d90: 8fc3001c lw v1,28(s8)

401d94: 00000000 nop

401d98: 24630001 addiu v1,v1,1

401d9c: afc3001c sw v1,28(s8)

401da0: 8fc30018 lw v1,24(s8)

401da4: 00000000 nop

401da8: 24630001 addiu v1,v1,1

401dac: afc30018 sw v1,24(s8)

401db0: 10400005 beqz v0,401dc8 <strings\_not\_equal+0xd0>

401db4: 00000000 nop

401db8: 24020001 li v0,1

401dbc: 10000009 b 401de4 <strings\_not\_equal+0xec>

401dc0: 00000000 nop

401dc4: 00000000 nop

401dc8: 8fc2001c lw v0,28(s8)

401dcc: 00000000 nop

401dd0: 80420000 lb v0,0(v0)

401dd4: 00000000 nop

401dd8: 1440ffe3 bnez v0,401d68 <strings\_not\_equal+0x70>

401ddc: 00000000 nop

401de0: 00001021 move v0,zero

401de4: 03c0e821 move sp,s8

401de8: 8fbf002c lw ra,44(sp)

401dec: 8fbe0028 lw s8,40(sp)

401df0: 27bd0030 addiu sp,sp,48

401df4: 03e00008 jr ra

401df8: 00000000 nop

00401dfc <initialize\_bomb>:

401dfc: 27bdffe0 addiu sp,sp,-32

401e00: afbf001c sw ra,28(sp)

401e04: afbe0018 sw s8,24(sp)

401e08: 03a0f021 move s8,sp

401e0c: 3c1c0042 lui gp,0x42

401e10: 279cb190 addiu gp,gp,-20080

401e14: afbc0010 sw gp,16(sp)

401e18: 3c020040 lui v0,0x40

401e1c: 24421a88 addiu v0,v0,6792

401e20: 24040002 li a0,2

401e24: 00402821 move a1,v0

401e28: 8f828090 lw v0,-32624(gp)

401e2c: 00000000 nop

401e30: 0040c821 move t9,v0

401e34: 0320f809 jalr t9

401e38: 00000000 nop

401e3c: 8fdc0010 lw gp,16(s8)

401e40: 03c0e821 move sp,s8

401e44: 8fbf001c lw ra,28(sp)

401e48: 8fbe0018 lw s8,24(sp)

401e4c: 27bd0020 addiu sp,sp,32

401e50: 03e00008 jr ra

401e54: 00000000 nop

00401e58 <blank\_line>:

401e58: 27bdffd8 addiu sp,sp,-40

401e5c: afbf0024 sw ra,36(sp)

401e60: afbe0020 sw s8,32(sp)

401e64: 03a0f021 move s8,sp

401e68: 3c1c0042 lui gp,0x42

401e6c: 279cb190 addiu gp,gp,-20080

401e70: afbc0010 sw gp,16(sp)

401e74: afc40028 sw a0,40(s8)

401e78: 8fc20028 lw v0,40(s8)

401e7c: 00000000 nop

401e80: afc20018 sw v0,24(s8)

401e84: 1000001c b 401ef8 <blank\_line+0xa0>

401e88: 00000000 nop

401e8c: 8f828064 lw v0,-32668(gp)

401e90: 00000000 nop

401e94: 0040c821 move t9,v0

401e98: 0320f809 jalr t9

401e9c: 00000000 nop

401ea0: 8fdc0010 lw gp,16(s8)

401ea4: 8c430000 lw v1,0(v0)

401ea8: 8fc20018 lw v0,24(s8)

401eac: 00000000 nop

401eb0: 80420000 lb v0,0(v0)

401eb4: 00000000 nop

401eb8: 00021040 sll v0,v0,0x1

401ebc: 00621021 addu v0,v1,v0

401ec0: 94420000 lhu v0,0(v0)

401ec4: 00000000 nop

401ec8: 30422000 andi v0,v0,0x2000

401ecc: 2c420001 sltiu v0,v0,1

401ed0: 304200ff andi v0,v0,0xff

401ed4: 8fc30018 lw v1,24(s8)

401ed8: 00000000 nop

401edc: 24630001 addiu v1,v1,1

401ee0: afc30018 sw v1,24(s8)

401ee4: 10400004 beqz v0,401ef8 <blank\_line+0xa0>

401ee8: 00000000 nop

401eec: 00001021 move v0,zero

401ef0: 10000008 b 401f14 <blank\_line+0xbc>

401ef4: 00000000 nop

401ef8: 8fc20018 lw v0,24(s8)

401efc: 00000000 nop

401f00: 80420000 lb v0,0(v0)

401f04: 00000000 nop

401f08: 1440ffe0 bnez v0,401e8c <blank\_line+0x34>

401f0c: 00000000 nop

401f10: 24020001 li v0,1

401f14: 03c0e821 move sp,s8

401f18: 8fbf0024 lw ra,36(sp)

401f1c: 8fbe0020 lw s8,32(sp)

401f20: 27bd0028 addiu sp,sp,40

401f24: 03e00008 jr ra

401f28: 00000000 nop

00401f2c <skip>:

401f2c: 27bdffd8 addiu sp,sp,-40

401f30: afbf0024 sw ra,36(sp)

401f34: afbe0020 sw s8,32(sp)

401f38: 03a0f021 move s8,sp

401f3c: 3c1c0042 lui gp,0x42

401f40: 279cb190 addiu gp,gp,-20080

401f44: afbc0010 sw gp,16(sp)

401f48: 10000002 b 401f54 <skip+0x28>

401f4c: 00000000 nop

401f50: 00000000 nop

401f54: 8f838058 lw v1,-32680(gp)

401f58: 3c020041 lui v0,0x41

401f5c: 8c423240 lw v0,12864(v0)

401f60: 00000000 nop

401f64: 00021100 sll v0,v0,0x4

401f68: 00022080 sll a0,v0,0x2

401f6c: 00441021 addu v0,v0,a0

401f70: 00621821 addu v1,v1,v0

401f74: 8f828050 lw v0,-32688(gp)

401f78: 00000000 nop

401f7c: 8c420000 lw v0,0(v0)

401f80: 00602021 move a0,v1

401f84: 24050050 li a1,80

401f88: 00403021 move a2,v0

401f8c: 8f82807c lw v0,-32644(gp)

401f90: 00000000 nop

401f94: 0040c821 move t9,v0

401f98: 0320f809 jalr t9

401f9c: 00000000 nop

401fa0: 8fdc0010 lw gp,16(s8)

401fa4: afc20018 sw v0,24(s8)

401fa8: 8fc20018 lw v0,24(s8)

401fac: 00000000 nop

401fb0: 10400007 beqz v0,401fd0 <skip+0xa4>

401fb4: 00000000 nop

401fb8: 8fc40018 lw a0,24(s8)

401fbc: 0c100796 jal 401e58 <blank\_line>

401fc0: 00000000 nop

401fc4: 8fdc0010 lw gp,16(s8)

401fc8: 1440ffe1 bnez v0,401f50 <skip+0x24>

401fcc: 00000000 nop

401fd0: 8fc20018 lw v0,24(s8)

401fd4: 03c0e821 move sp,s8

401fd8: 8fbf0024 lw ra,36(sp)

401fdc: 8fbe0020 lw s8,32(sp)

401fe0: 27bd0028 addiu sp,sp,40

401fe4: 03e00008 jr ra

401fe8: 00000000 nop

00401fec <read\_line>:

401fec: 27bdffd8 addiu sp,sp,-40

401ff0: afbf0024 sw ra,36(sp)

401ff4: afbe0020 sw s8,32(sp)

401ff8: 03a0f021 move s8,sp

401ffc: 3c1c0042 lui gp,0x42

402000: 279cb190 addiu gp,gp,-20080

402004: afbc0010 sw gp,16(sp)

402008: 0c1007cb jal 401f2c <skip>

40200c: 00000000 nop

402010: 8fdc0010 lw gp,16(s8)

402014: afc2001c sw v0,28(s8)

402018: 8fc2001c lw v0,28(s8)

40201c: 00000000 nop

402020: 14400033 bnez v0,4020f0 <read\_line+0x104>

402024: 00000000 nop

402028: 8f828050 lw v0,-32688(gp)

40202c: 00000000 nop

402030: 8c430000 lw v1,0(v0)

402034: 8f82804c lw v0,-32692(gp)

402038: 00000000 nop

40203c: 8c420000 lw v0,0(v0)

402040: 00000000 nop

402044: 1062001b beq v1,v0,4020b4 <read\_line+0xc8>

402048: 00000000 nop

40204c: 3c020040 lui v0,0x40

402050: 2444283c addiu a0,v0,10300

402054: 8f828080 lw v0,-32640(gp)

402058: 00000000 nop

40205c: 0040c821 move t9,v0

402060: 0320f809 jalr t9

402064: 00000000 nop

402068: 8fdc0010 lw gp,16(s8)

40206c: 10400007 beqz v0,40208c <read\_line+0xa0>

402070: 00000000 nop

402074: 00002021 move a0,zero

402078: 8f828034 lw v0,-32716(gp)

40207c: 00000000 nop

402080: 0040c821 move t9,v0

402084: 0320f809 jalr t9

402088: 00000000 nop

40208c: 8f82804c lw v0,-32692(gp)

402090: 00000000 nop

402094: 8c430000 lw v1,0(v0)

402098: 8f828050 lw v0,-32688(gp)

40209c: 00000000 nop

4020a0: ac430000 sw v1,0(v0)

4020a4: 0c1007cb jal 401f2c <skip>

4020a8: 00000000 nop

4020ac: 8fdc0010 lw gp,16(s8)

4020b0: afc2001c sw v0,28(s8)

4020b4: 8fc2001c lw v0,28(s8)

4020b8: 00000000 nop

4020bc: 1440000c bnez v0,4020f0 <read\_line+0x104>

4020c0: 00000000 nop

4020c4: 3c020040 lui v0,0x40

4020c8: 24442848 addiu a0,v0,10312

4020cc: 8f828038 lw v0,-32712(gp)

4020d0: 00000000 nop

4020d4: 0040c821 move t9,v0

4020d8: 0320f809 jalr t9

4020dc: 00000000 nop

4020e0: 8fdc0010 lw gp,16(s8)

4020e4: 0c10087c jal 4021f0 <explode\_bomb>

4020e8: 00000000 nop

4020ec: 8fdc0010 lw gp,16(s8)

4020f0: afc00018 sw zero,24(s8)

4020f4: 10000005 b 40210c <read\_line+0x120>

4020f8: 00000000 nop

4020fc: 8fc20018 lw v0,24(s8)

402100: 00000000 nop

402104: 24420001 addiu v0,v0,1

402108: afc20018 sw v0,24(s8)

40210c: 8fc20018 lw v0,24(s8)

402110: 8fc3001c lw v1,28(s8)

402114: 00000000 nop

402118: 00621021 addu v0,v1,v0

40211c: 80420000 lb v0,0(v0)

402120: 00000000 nop

402124: 1440fff5 bnez v0,4020fc <read\_line+0x110>

402128: 00000000 nop

40212c: 8fc30018 lw v1,24(s8)

402130: 2402004f li v0,79

402134: 1462000c bne v1,v0,402168 <read\_line+0x17c>

402138: 00000000 nop

40213c: 3c020040 lui v0,0x40

402140: 24442868 addiu a0,v0,10344

402144: 8f828038 lw v0,-32712(gp)

402148: 00000000 nop

40214c: 0040c821 move t9,v0

402150: 0320f809 jalr t9

402154: 00000000 nop

402158: 8fdc0010 lw gp,16(s8)

40215c: 0c10087c jal 4021f0 <explode\_bomb>

402160: 00000000 nop

402164: 8fdc0010 lw gp,16(s8)

402168: 3c020041 lui v0,0x41

40216c: 8c423240 lw v0,12864(v0)

402170: 8fc30018 lw v1,24(s8)

402174: 00000000 nop

402178: 2464ffff addiu a0,v1,-1

40217c: 8f838058 lw v1,-32680(gp)

402180: 00021100 sll v0,v0,0x4

402184: 00022880 sll a1,v0,0x2

402188: 00451021 addu v0,v0,a1

40218c: 00441021 addu v0,v0,a0

402190: 00621021 addu v0,v1,v0

402194: a0400000 sb zero,0(v0)

402198: 8f838058 lw v1,-32680(gp)

40219c: 3c020041 lui v0,0x41

4021a0: 8c423240 lw v0,12864(v0)

4021a4: 00000000 nop

4021a8: 00021100 sll v0,v0,0x4

4021ac: 00022080 sll a0,v0,0x2

4021b0: 00441021 addu v0,v0,a0

4021b4: 00621021 addu v0,v1,v0

4021b8: afc2001c sw v0,28(s8)

4021bc: 3c020041 lui v0,0x41

4021c0: 8c423240 lw v0,12864(v0)

4021c4: 00000000 nop

4021c8: 24430001 addiu v1,v0,1

4021cc: 3c020041 lui v0,0x41

4021d0: ac433240 sw v1,12864(v0)

4021d4: 8fc2001c lw v0,28(s8)

4021d8: 03c0e821 move sp,s8

4021dc: 8fbf0024 lw ra,36(sp)

4021e0: 8fbe0020 lw s8,32(sp)

4021e4: 27bd0028 addiu sp,sp,40

4021e8: 03e00008 jr ra

4021ec: 00000000 nop

004021f0 <explode\_bomb>:

4021f0: 27bdffe0 addiu sp,sp,-32

4021f4: afbf001c sw ra,28(sp)

4021f8: afbe0018 sw s8,24(sp)

4021fc: 03a0f021 move s8,sp

402200: 3c1c0042 lui gp,0x42

402204: 279cb190 addiu gp,gp,-20080

402208: afbc0010 sw gp,16(sp)

40220c: 3c020040 lui v0,0x40

402210: 24442884 addiu a0,v0,10372

402214: 8f828038 lw v0,-32712(gp)

402218: 00000000 nop

40221c: 0040c821 move t9,v0

402220: 0320f809 jalr t9

402224: 00000000 nop

402228: 8fdc0010 lw gp,16(s8)

40222c: 3c020040 lui v0,0x40

402230: 24442890 addiu a0,v0,10384

402234: 8f828038 lw v0,-32712(gp)

402238: 00000000 nop

40223c: 0040c821 move t9,v0

402240: 0320f809 jalr t9

402244: 00000000 nop

402248: 8fdc0010 lw gp,16(s8)

40224c: 24040008 li a0,8

402250: 8f828034 lw v0,-32716(gp)

402254: 00000000 nop

402258: 0040c821 move t9,v0

40225c: 0320f809 jalr t9

402260: 00000000 nop

00402264 <phase\_defused>:

402264: 27bdff88 addiu sp,sp,-120

402268: afbf0074 sw ra,116(sp)

40226c: afbe0070 sw s8,112(sp)

402270: 03a0f021 move s8,sp

402274: 3c1c0042 lui gp,0x42

402278: 279cb190 addiu gp,gp,-20080

40227c: afbc0010 sw gp,16(sp)

402280: 3c020041 lui v0,0x41

402284: 8c433240 lw v1,12864(v0)

402288: 24020006 li v0,6

40228c: 14620039 bne v1,v0,402374 <phase\_defused+0x110>

402290: 00000000 nop

402294: 8f828058 lw v0,-32680(gp)

402298: 00000000 nop

40229c: 244400f0 addiu a0,v0,240

4022a0: 3c020040 lui v0,0x40

4022a4: 244328a8 addiu v1,v0,10408

4022a8: 27c20068 addiu v0,s8,104

4022ac: 00602821 move a1,v1

4022b0: 00403021 move a2,v0

4022b4: 27c20018 addiu v0,s8,24

4022b8: 00403821 move a3,v0

4022bc: 8f828084 lw v0,-32636(gp)

4022c0: 00000000 nop

4022c4: 0040c821 move t9,v0

4022c8: 0320f809 jalr t9

4022cc: 00000000 nop

4022d0: 8fdc0010 lw gp,16(s8)

4022d4: 00401821 move v1,v0

4022d8: 24020002 li v0,2

4022dc: 1462001d bne v1,v0,402354 <phase\_defused+0xf0>

4022e0: 00000000 nop

4022e4: 27c20018 addiu v0,s8,24

4022e8: 00402021 move a0,v0

4022ec: 3c020040 lui v0,0x40

4022f0: 244528b0 addiu a1,v0,10416

4022f4: 0c10073e jal 401cf8 <strings\_not\_equal>

4022f8: 00000000 nop

4022fc: 8fdc0010 lw gp,16(s8)

402300: 14400014 bnez v0,402354 <phase\_defused+0xf0>

402304: 00000000 nop

402308: 3c020040 lui v0,0x40

40230c: 244428c0 addiu a0,v0,10432

402310: 8f828038 lw v0,-32712(gp)

402314: 00000000 nop

402318: 0040c821 move t9,v0

40231c: 0320f809 jalr t9

402320: 00000000 nop

402324: 8fdc0010 lw gp,16(s8)

402328: 3c020040 lui v0,0x40

40232c: 244428e8 addiu a0,v0,10472

402330: 8f828038 lw v0,-32712(gp)

402334: 00000000 nop

402338: 0040c821 move t9,v0

40233c: 0320f809 jalr t9

402340: 00000000 nop

402344: 8fdc0010 lw gp,16(s8)

402348: 0c100664 jal 401990 <secret\_phase>

40234c: 00000000 nop

402350: 8fdc0010 lw gp,16(s8)

402354: 3c020040 lui v0,0x40

402358: 24442920 addiu a0,v0,10528

40235c: 8f828038 lw v0,-32712(gp)

402360: 00000000 nop

402364: 0040c821 move t9,v0

402368: 0320f809 jalr t9

40236c: 00000000 nop

402370: 8fdc0010 lw gp,16(s8)

402374: 03c0e821 move sp,s8

402378: 8fbf0074 lw ra,116(sp)

40237c: 8fbe0070 lw s8,112(sp)

402380: 27bd0078 addiu sp,sp,120

402384: 03e00008 jr ra

402388: 00000000 nop

40238c: 00000000 nop

00402390 <\_\_libc\_csu\_fini>:

402390: 03e00008 jr ra

402394: 00000000 nop

00402398 <\_\_libc\_csu\_init>:

402398: 3c1c0002 lui gp,0x2

40239c: 279c8df8 addiu gp,gp,-29192

4023a0: 0399e021 addu gp,gp,t9

4023a4: 27bdffc8 addiu sp,sp,-56

4023a8: afbf0034 sw ra,52(sp)

4023ac: afb50030 sw s5,48(sp)

4023b0: afb4002c sw s4,44(sp)

4023b4: afb30028 sw s3,40(sp)

4023b8: afb20024 sw s2,36(sp)

4023bc: afb10020 sw s1,32(sp)

4023c0: afb0001c sw s0,28(sp)

4023c4: afbc0010 sw gp,16(sp)

4023c8: 8f99802c lw t9,-32724(gp)

4023cc: 00809821 move s3,a0

4023d0: 00a0a021 move s4,a1

4023d4: 0320f809 jalr t9

4023d8: 00c0a821 move s5,a2

4023dc: 8fbc0010 lw gp,16(sp)

4023e0: 8f91801c lw s1,-32740(gp)

4023e4: 8f92801c lw s2,-32740(gp)

4023e8: 02519023 subu s2,s2,s1

4023ec: 00129083 sra s2,s2,0x2

4023f0: 1240000a beqz s2,40241c <\_\_libc\_csu\_init+0x84>

4023f4: 00008021 move s0,zero

4023f8: 8e390000 lw t9,0(s1)

4023fc: 26100001 addiu s0,s0,1

402400: 02602021 move a0,s3

402404: 02802821 move a1,s4

402408: 0320f809 jalr t9

40240c: 02a03021 move a2,s5

402410: 0212102b sltu v0,s0,s2

402414: 1440fff8 bnez v0,4023f8 <\_\_libc\_csu\_init+0x60>

402418: 26310004 addiu s1,s1,4

40241c: 8fbf0034 lw ra,52(sp)

402420: 8fb50030 lw s5,48(sp)

402424: 8fb4002c lw s4,44(sp)

402428: 8fb30028 lw s3,40(sp)

40242c: 8fb20024 lw s2,36(sp)

402430: 8fb10020 lw s1,32(sp)

402434: 8fb0001c lw s0,28(sp)

402438: 03e00008 jr ra

40243c: 27bd0038 addiu sp,sp,56

00402440 <\_\_do\_global\_ctors\_aux>:

402440: 3c020041 lui v0,0x41

402444: 8c593058 lw t9,12376(v0)

402448: 27bdffd8 addiu sp,sp,-40

40244c: 2402ffff li v0,-1

402450: afbf0024 sw ra,36(sp)

402454: afb10020 sw s1,32(sp)

402458: 13220009 beq t9,v0,402480 <\_\_do\_global\_ctors\_aux+0x40>

40245c: afb0001c sw s0,28(sp)

402460: 3c100041 lui s0,0x41

402464: 26103058 addiu s0,s0,12376

402468: 2411ffff li s1,-1

40246c: 0320f809 jalr t9

402470: 2610fffc addiu s0,s0,-4

402474: 8e190000 lw t9,0(s0)

402478: 1731fffc bne t9,s1,40246c <\_\_do\_global\_ctors\_aux+0x2c>

40247c: 00000000 nop

402480: 8fbf0024 lw ra,36(sp)

402484: 8fb10020 lw s1,32(sp)

402488: 8fb0001c lw s0,28(sp)

40248c: 03e00008 jr ra

402490: 27bd0028 addiu sp,sp,40

...

Disassembly of section .MIPS.stubs:

004024a0 <.MIPS.stubs>:

4024a0: 8f998010 lw t9,-32752(gp)

4024a4: 03e07821 move t7,ra

4024a8: 0320f809 jalr t9

4024ac: 2418001d li t8,29

4024b0: 8f998010 lw t9,-32752(gp)

4024b4: 03e07821 move t7,ra

4024b8: 0320f809 jalr t9

4024bc: 2418001a li t8,26

4024c0: 8f998010 lw t9,-32752(gp)

4024c4: 03e07821 move t7,ra

4024c8: 0320f809 jalr t9

4024cc: 24180019 li t8,25

4024d0: 8f998010 lw t9,-32752(gp)

4024d4: 03e07821 move t7,ra

4024d8: 0320f809 jalr t9

4024dc: 24180018 li t8,24

4024e0: 8f998010 lw t9,-32752(gp)

4024e4: 03e07821 move t7,ra

4024e8: 0320f809 jalr t9

4024ec: 24180017 li t8,23

4024f0: 8f998010 lw t9,-32752(gp)

4024f4: 03e07821 move t7,ra

4024f8: 0320f809 jalr t9

4024fc: 24180015 li t8,21

402500: 8f998010 lw t9,-32752(gp)

402504: 03e07821 move t7,ra

402508: 0320f809 jalr t9

40250c: 24180013 li t8,19

402510: 8f998010 lw t9,-32752(gp)

402514: 03e07821 move t7,ra

402518: 0320f809 jalr t9

40251c: 24180012 li t8,18

402520: 8f998010 lw t9,-32752(gp)

402524: 03e07821 move t7,ra

402528: 0320f809 jalr t9

40252c: 24180011 li t8,17

402530: 8f998010 lw t9,-32752(gp)

402534: 03e07821 move t7,ra

402538: 0320f809 jalr t9

40253c: 2418000e li t8,14

402540: 8f998010 lw t9,-32752(gp)

402544: 03e07821 move t7,ra

402548: 0320f809 jalr t9

40254c: 2418000b li t8,11

402550: 8f998010 lw t9,-32752(gp)

402554: 03e07821 move t7,ra

402558: 0320f809 jalr t9

40255c: 2418000a li t8,10

402560: 8f998010 lw t9,-32752(gp)

402564: 03e07821 move t7,ra

402568: 0320f809 jalr t9

40256c: 24180009 li t8,9

402570: 8f998010 lw t9,-32752(gp)

402574: 03e07821 move t7,ra

402578: 0320f809 jalr t9

40257c: 24180007 li t8,7

402580: 8f998010 lw t9,-32752(gp)

402584: 03e07821 move t7,ra

402588: 0320f809 jalr t9

40258c: 24180006 li t8,6

...

Disassembly of section .fini:

004025a0 <\_fini>:

4025a0: 3c1c0002 lui gp,0x2

4025a4: 279c8bf0 addiu gp,gp,-29712

4025a8: 0399e021 addu gp,gp,t9

4025ac: 27bdffe0 addiu sp,sp,-32

4025b0: afbf001c sw ra,28(sp)

4025b4: afbc0010 sw gp,16(sp)

4025b8: 04110001 bal 4025c0 <\_fini+0x20>

4025bc: 00000000 nop

4025c0: 3c1c0042 lui gp,0x42

4025c4: 279cb190 addiu gp,gp,-20080

4025c8: 8f998018 lw t9,-32744(gp)

4025cc: 27390830 addiu t9,t9,2096

4025d0: 0411f897 bal 400830 <\_\_do\_global\_dtors\_aux>

4025d4: 00000000 nop

4025d8: 8fbf001c lw ra,28(sp)

4025dc: 03e00008 jr ra

4025e0: 27bd0020 addiu sp,sp,32