# 64位版本

0000000000006da <fun>:

6da: 48 83 ec 68 sub $0x68,%rsp

6de: 64 48 8b 04 25 28 00 mov %fs:0x28,%rax

6e5: 00 00

6e7: 48 89 44 24 58 mov %rax,0x58(%rsp)

6ec: 31 c0 xor %eax,%eax

6ee: 89 3c 24 mov %edi,(%rsp)

6f1: 89 74 24 04 mov %esi,0x4(%rsp)

6f5: 89 54 24 08 mov %edx,0x8(%rsp)

6f9: 89 4c 24 0c mov %ecx,0xc(%rsp)

6fd: 44 89 44 24 10 mov %r8d,0x10(%rsp)

702: 44 89 4c 24 14 mov %r9d,0x14(%rsp)

707: 8b 44 24 70 mov 0x70(%rsp),%eax

70b: 89 44 24 18 mov %eax,0x18(%rsp)

70f: 8b 44 24 78 mov 0x78(%rsp),%eax

713: 89 44 24 1c mov %eax,0x1c(%rsp)

717: 8b 84 24 80 00 00 00 mov 0x80(%rsp),%eax

71e: 89 44 24 20 mov %eax,0x20(%rsp)

722: 8b 84 24 88 00 00 00 mov 0x88(%rsp),%eax

729: 89 44 24 24 mov %eax,0x24(%rsp)

72d: c7 44 24 28 00 00 00 movl $0x0,0x28(%rsp)

734: 00

735: b8 00 00 00 00 mov $0x0,%eax

73a: eb 0d jmp 749 <fun+0x6f>

73c: 48 63 d0 movslq %eax,%rdx

73f: 8b 14 94 mov (%rsp,%rdx,4),%edx

742: 01 54 24 28 add %edx,0x28(%rsp)

746: 83 c0 01 add $0x1,%eax

749: 83 f8 09 cmp $0x9,%eax

74c: 7e ee jle 73c <fun+0x62>

74e: 8b 44 24 28 mov 0x28(%rsp),%eax

752: 48 8b 4c 24 58 mov 0x58(%rsp),%rcx

757: 64 48 33 0c 25 28 00 xor %fs:0x28,%rcx

75e: 00 00

760: 75 05 jne 767 <fun+0x8d>

762: 48 83 c4 68 add $0x68,%rsp

766: c3 retq

767: e8 34 fe ff ff callq 5a0 <\_\_stack\_chk\_fail@plt>

000000000000076c <main>:

76c: 48 83 ec 08 sub $0x8,%rsp

770: 6a 0a pushq $0xa

772: 6a 09 pushq $0x9

774: 6a 08 pushq $0x8

776: 6a 07 pushq $0x7

778: 41 b9 06 00 00 00 mov $0x6,%r9d

77e: 41 b8 05 00 00 00 mov $0x5,%r8d

784: b9 04 00 00 00 mov $0x4,%ecx

789: ba 03 00 00 00 mov $0x3,%edx

78e: be 02 00 00 00 mov $0x2,%esi

793: bf 01 00 00 00 mov $0x1,%edi

798: e8 3d ff ff ff callq 6da <fun>

79d: 48 83 c4 20 add $0x20,%rsp

7a1: 89 c2 mov %eax,%edx

7a3: 48 8d 35 aa 00 00 00 lea 0xaa(%rip),%rsi # 854 <\_IO\_stdin\_used+0x4>

7aa: bf 01 00 00 00 mov $0x1,%edi

7af: b8 00 00 00 00 mov $0x0,%eax

7b4: e8 f7 fd ff ff callq 5b0 <\_\_printf\_chk@plt>

7b9: b8 00 00 00 00 mov $0x0,%eax

7be: 48 83 c4 08 add $0x8,%rsp

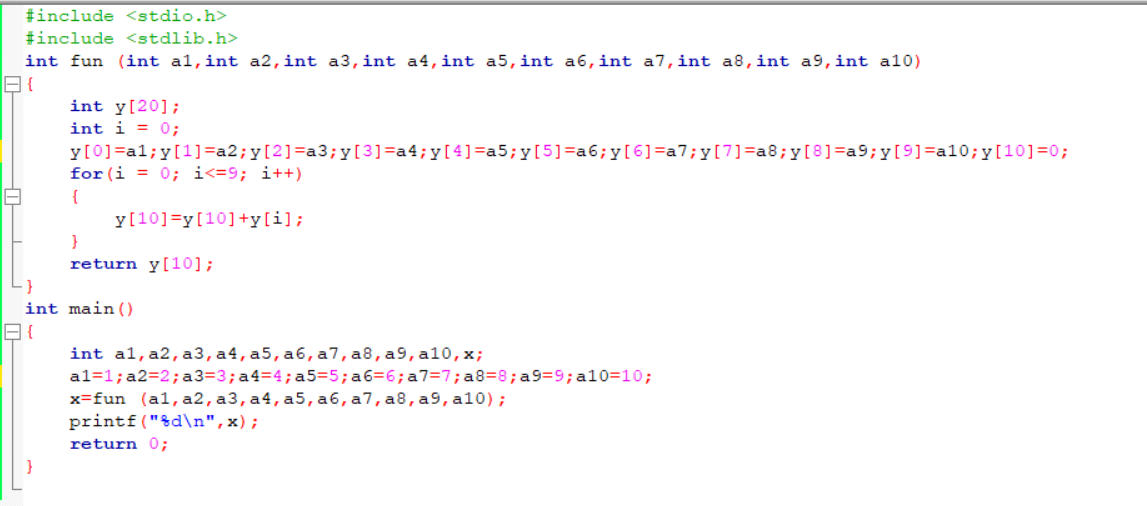
7c2: c3 retq

7c3: 66 2e 0f 1f 84 00 00 nopw %cs:0x0(%rax,%rax,1)

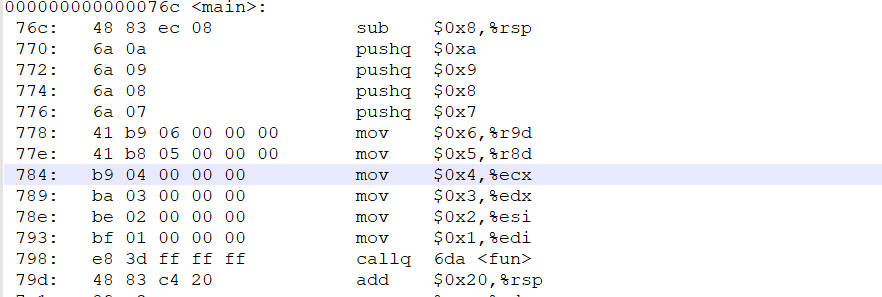
7ca: 00 00 00

7cd: 0f 1f 00 nopl (%rax)

**C代码截图**



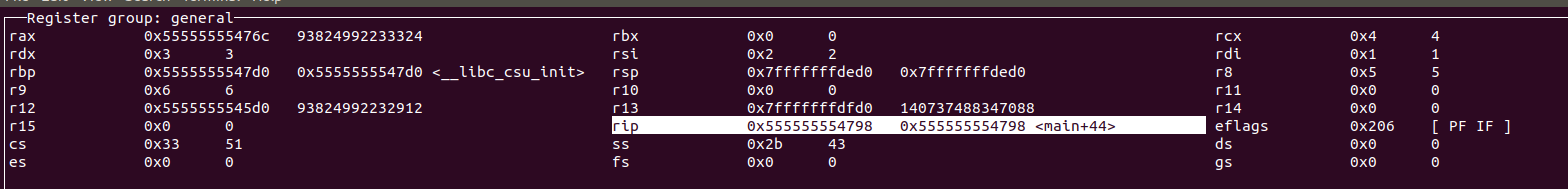
**分析如下：**

**先看到main函数**

**在callq 6da <fun>执行前，函数在传参过程，可见调用了六个寄存器进行传参数，如果参数多于6个，就用栈来传参，这里一共有四个变量需要调用栈，所以有四句的pushq命令。执行callq 66a <fun>的时候，栈指针rsp先减去8，腾出位置放返回地址此时栈框架如下**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
|  | **0x7** |
| **rsp** | **0x79d（返回地址）** |
|  | **小地址** |

**寄存器的值如下**



**Rsp如下**



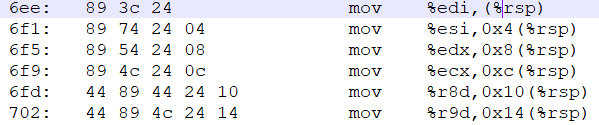
**进入fun函数**



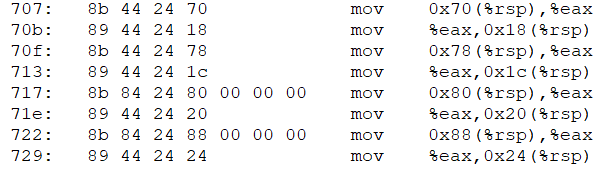
**首先将栈指针减小开辟出空间，此时rsp**



**然后将寄存器的值放入新开辟的栈中**



**然后将四个参数也移到栈里，用eax做中转**



**最后的累加值放在数组最后一位，事先清零**

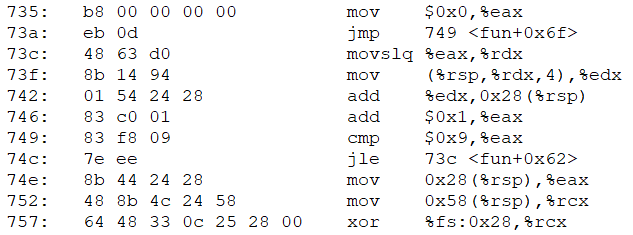


**此时的栈帧结构**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
|  | **0x7** |
|  | **0x79d（返回地址）** |
|  |  |
|  |  |
|  |  |
|  |  |
| **数组** | **数组的第十一个元素，用来存放计算得数 （）** |
| **数组** | **0xa第十个参数** |
| **数组** | **0x9第九个参数** |
| **数组** | **0x8第八个参数** |
| **数组** | **0x7第七个参数** |
| **数组** | **0x6第六个参数** |
| **数组** | **0x5第五个参数** |
| **数组** | **0x4第四个参数** |
| **数组** | **0x3第三个参数** |
| **数组** | **0x2第二个参数** |
| **Rsp** | **0x1 第一个参数** |
|  | **小地址** |
|  |  |

**进行循环累加操作操作，可见该过程用rsp的值作为数组的起始地址，rdx的值作为数组的索引值，得数放在数组的第十一位也就是0x28（%rsp），**

**最后赋值给eax返回。**

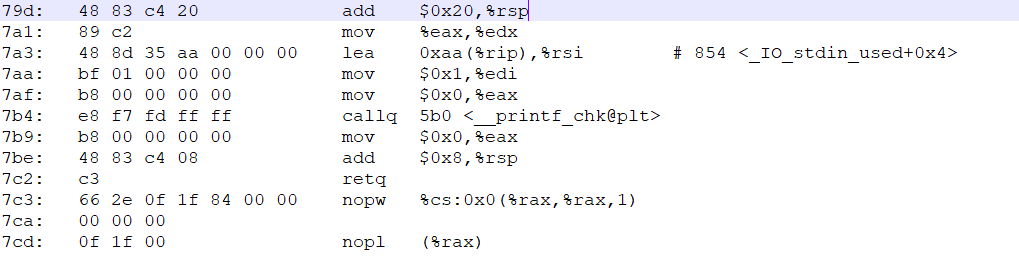


**返回前将栈指针减去原来加上的数**



**这时栈帧结构为**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **Main的返回** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
| **rsp** | **0x7** |
|  | **小地址** |

**返回后把rsp加上0x20，执行打印操作，返回，main结束。**

# 32位版本

0000059d <fun>:

59d: 83 ec 6c sub $0x6c,%esp

5a0: 65 a1 14 00 00 00 mov %gs:0x14,%eax

5a6: 89 44 24 5c mov %eax,0x5c(%esp)

5aa: 31 c0 xor %eax,%eax

5ac: 8b 44 24 70 mov 0x70(%esp),%eax

5b0: 89 44 24 0c mov %eax,0xc(%esp)

5b4: 8b 44 24 74 mov 0x74(%esp),%eax

5b8: 89 44 24 10 mov %eax,0x10(%esp)

5bc: 8b 44 24 78 mov 0x78(%esp),%eax

5c0: 89 44 24 14 mov %eax,0x14(%esp)

5c4: 8b 44 24 7c mov 0x7c(%esp),%eax

5c8: 89 44 24 18 mov %eax,0x18(%esp)

5cc: 8b 84 24 80 00 00 00 mov 0x80(%esp),%eax

5d3: 89 44 24 1c mov %eax,0x1c(%esp)

5d7: 8b 84 24 84 00 00 00 mov 0x84(%esp),%eax

5de: 89 44 24 20 mov %eax,0x20(%esp)

5e2: 8b 84 24 88 00 00 00 mov 0x88(%esp),%eax

5e9: 89 44 24 24 mov %eax,0x24(%esp)

5ed: 8b 84 24 8c 00 00 00 mov 0x8c(%esp),%eax

5f4: 89 44 24 28 mov %eax,0x28(%esp)

5f8: 8b 84 24 90 00 00 00 mov 0x90(%esp),%eax

5ff: 89 44 24 2c mov %eax,0x2c(%esp)

603: 8b 84 24 94 00 00 00 mov 0x94(%esp),%eax

60a: 89 44 24 30 mov %eax,0x30(%esp)

60e: c7 44 24 34 00 00 00 movl $0x0,0x34(%esp)

615: 00

616: b8 00 00 00 00 mov $0x0,%eax

61b: eb 0b jmp 628 <fun+0x8b>

61d: 8b 54 84 0c mov 0xc(%esp,%eax,4),%edx

621: 01 54 24 34 add %edx,0x34(%esp)

625: 83 c0 01 add $0x1,%eax

628: 83 f8 09 cmp $0x9,%eax

62b: 7e f0 jle 61d <fun+0x80>

62d: 8b 44 24 34 mov 0x34(%esp),%eax

631: 8b 4c 24 5c mov 0x5c(%esp),%ecx

635: 65 33 0d 14 00 00 00 xor %gs:0x14,%ecx

63c: 75 04 jne 642 <fun+0xa5>

63e: 83 c4 6c add $0x6c,%esp

641: c3 ret

642: e8 d9 00 00 00 call 720 <\_\_stack\_chk\_fail\_local>

00000647 <main>:

647: 8d 4c 24 04 lea 0x4(%esp),%ecx

64b: 83 e4 f0 and $0xfffffff0,%esp

64e: ff 71 fc pushl -0x4(%ecx)

651: 55 push %ebp

652: 89 e5 mov %esp,%ebp

654: 53 push %ebx

655: 51 push %ecx

656: e8 45 fe ff ff call 4a0 <\_\_x86.get\_pc\_thunk.bx>

65b: 81 c3 79 19 00 00 add $0x1979,%ebx

661: 83 ec 08 sub $0x8,%esp

664: 6a 0a push $0xa

666: 6a 09 push $0x9

668: 6a 08 push $0x8

66a: 6a 07 push $0x7

66c: 6a 06 push $0x6

66e: 6a 05 push $0x5

670: 6a 04 push $0x4

672: 6a 03 push $0x3

674: 6a 02 push $0x2

676: 6a 01 push $0x1

678: e8 20 ff ff ff call 59d <fun>

67d: 83 c4 2c add $0x2c,%esp

680: 50 push %eax

681: 8d 83 7c e7 ff ff lea -0x1884(%ebx),%eax

687: 50 push %eax

688: 6a 01 push $0x1

68a: e8 b1 fd ff ff call 440 <\_\_printf\_chk@plt>

68f: 83 c4 10 add $0x10,%esp

692: b8 00 00 00 00 mov $0x0,%eax

697: 8d 65 f8 lea -0x8(%ebp),%esp

69a: 59 pop %ecx

69b: 5b pop %ebx

69c: 5d pop %ebp

69d: 8d 61 fc lea -0x4(%ecx),%esp

6a0: c3 ret

6a1: 66 90 xchg %ax,%ax

6a3: 66 90 xchg %ax,%ax

6a5: 66 90 xchg %ax,%ax

6a7: 66 90 xchg %ax,%ax

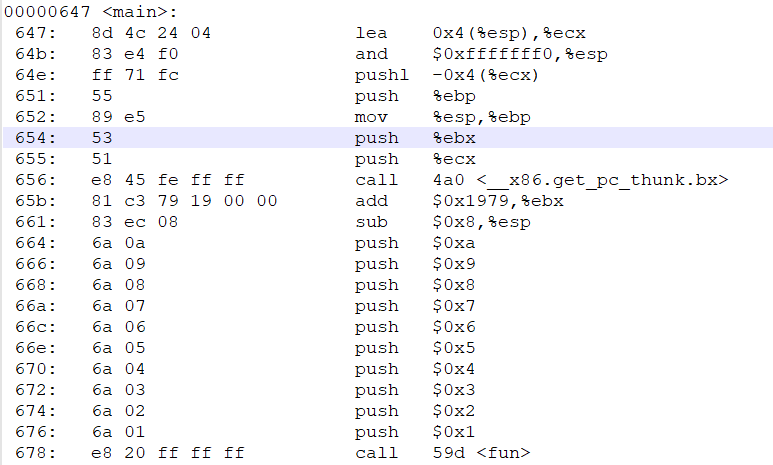
6a9: 66 90 xchg %ax,%ax

6ab: 66 90 xchg %ax,%ax

6ad: 66 90 xchg %ax,%ax

6af: 90 nop

**先看到main函数**

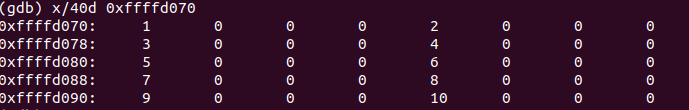




**先为main函数开辟初始的栈空间，将十个变量压入栈中，esp随之改变，然后call fun函数，此时esp：**



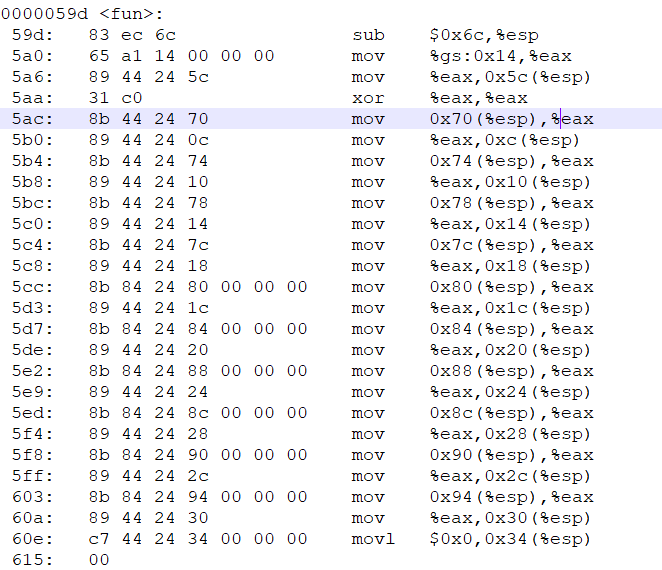
**压入的数据：**



**栈帧结构：**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
|  | **0x7** |
|  | **0x6** |
|  | **0x5** |
|  | **0x4** |
|  | **0x3** |
|  | **0x2** |
|  | **0x1** |
| **esp 0xffffd070** | **0x65b （返回地址）** |
|  | **小地址** |

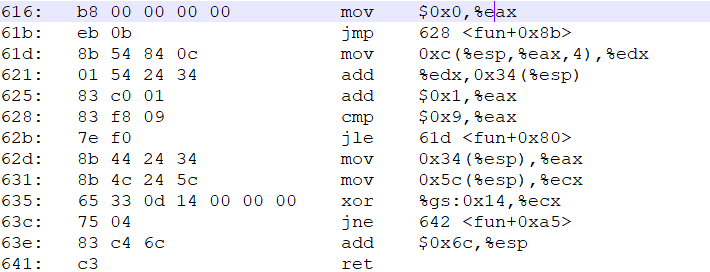
**fun部分，先将esp减掉0x6c为开辟空间**



**清零eax，然后用eax做中转，吧main函数中的值压到新开辟的栈里，相当于是向数组中传递元素。最后再吧0x34（esp）清零**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
|  | **0x7** |
|  | **0x6** |
|  | **0x5** |
|  | **0x4** |
|  | **0x3** |
|  | **0x2** |
|  | **0x1** |
|  | **0x65b （返回地址）** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| **数组** | **数组的第十一个元素，用来存放计算得数 （）** |
| **数组** | **0xa第十个参数** |
| **数组** | **0x9第九个参数** |
| **数组** | **0x8第八个参数** |
| **数组** | **0x7第七个参数** |
| **数组** | **0x6第六个参数** |
| **数组** | **0x5第五个参数** |
| **数组** | **0x4第四个参数** |
| **数组** | **0x3第三个参数** |
| **数组** | **0x2第二个参数** |
| **Esp+0xc数组** | **0x1 第一个参数** |
|  |  |
| **esp** |  |
|  | **小地址** |
|  |  |

**累加操作以后把答案保存在0x34（esp），可见该过程用0xc（esp）的值作为数组的起始地址，edx的值作为数组的索引值，可最后再赋值给eax返回**



**返回main后**

|  |  |
| --- | --- |
|  | **大地址** |
|  | **0xa** |
|  | **0x9** |
|  | **0x8** |
|  | **0x7** |
|  | **0x6** |
|  | **0x5** |
|  | **0x4** |
|  | **0x3** |
|  | **0x2** |
| **esp 0xffffd070** | **0x1** |
|  |  |
|  | **小地址** |

**返回main函数后，还原esp指针调用printf与64位无异。**

