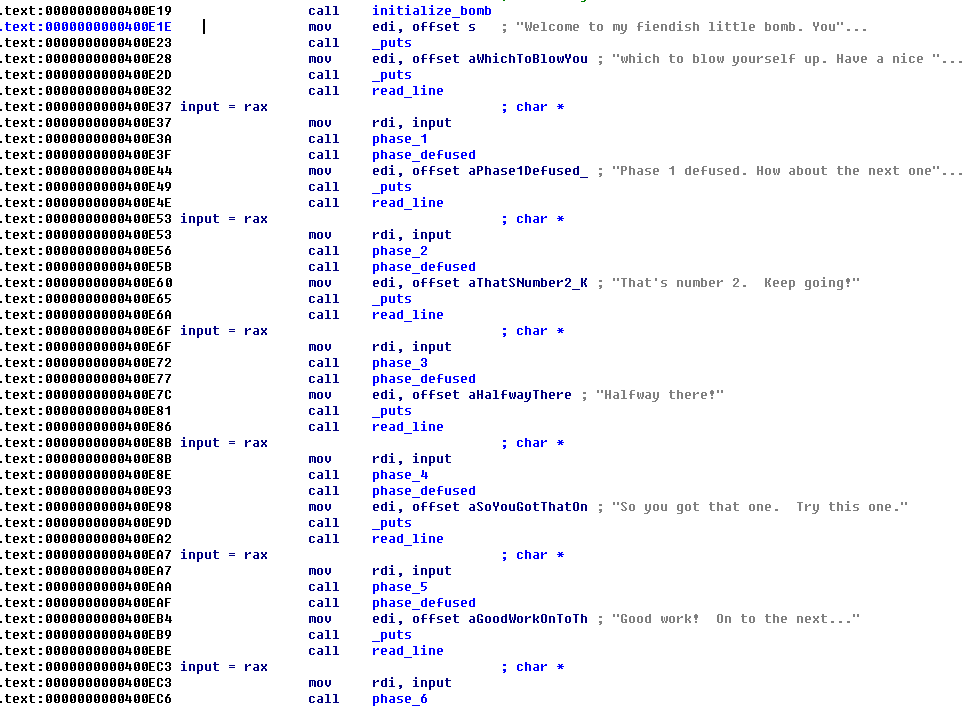
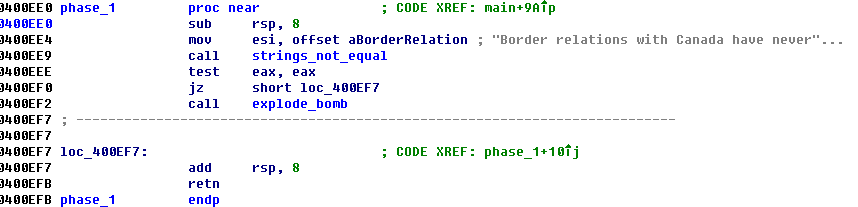
**Bomb-Lab实验**

实验给了一个可执行文件，要求构造一组输入避免程序爆炸顺利运行，在源文件中可以大概看出程序的运行逻辑。实验使用IDA实现反汇编，构造合适的字符串避免跳转至爆炸函数，使程序顺利运行。



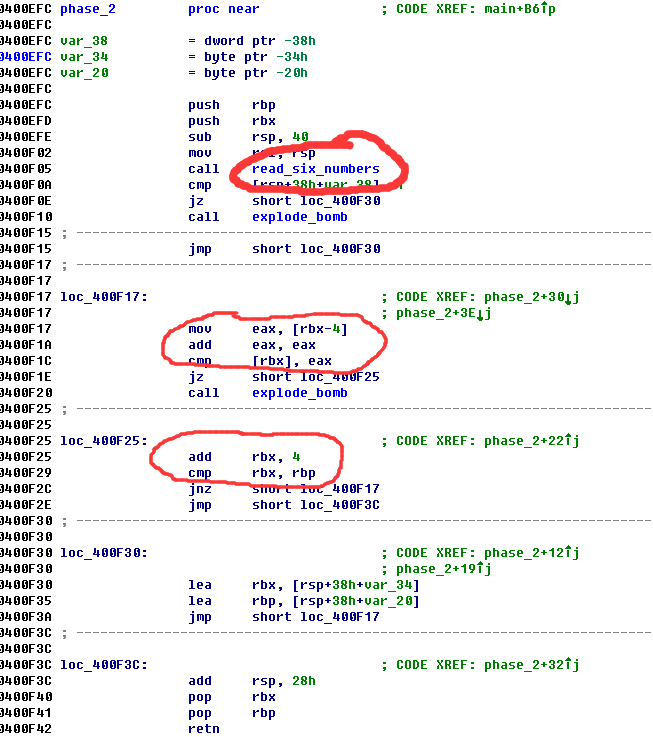
**Main函数**

主函数汇编如上图所示，结合汇编代码和程序运行结果可知需要在6个阶段分别输入合适的字符串，因此需要分别分析六个阶段对应的函数。



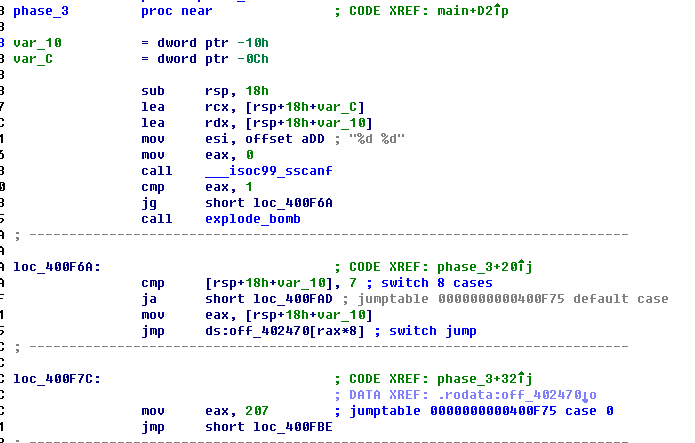
**Phase\_1**

第一阶段的函数如上图所示，要求输入与一个常量字符串相等，因此第一阶段的输入即为该字符串：Border relations with Canada have never been better.



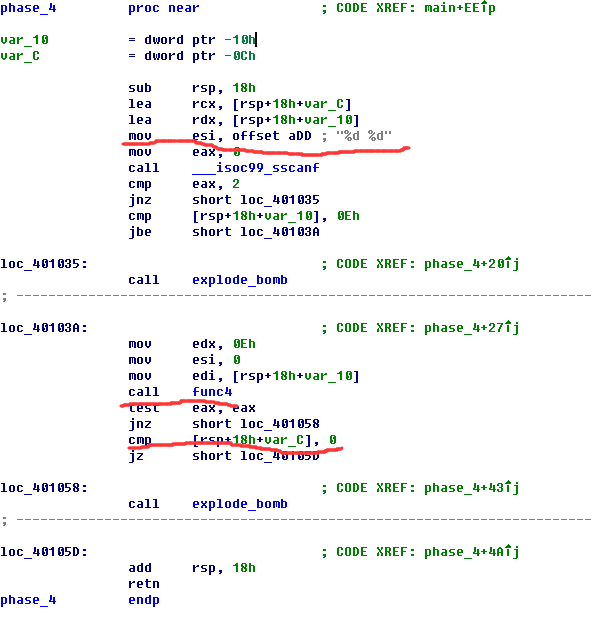
**Phase\_2**

第二阶段的代码如上图所示，第一个红圈中read\_six\_number函数需要读取六个数字，查看该函数可知要求的输入格式为空格隔开的六个数字，可以理解为数组，紧接着判断第一个数字是否为1，如果不是则爆炸。第三个红圈中为一个循环中的结尾运算和循环终止条件，rbx为当前数字的指针，rbp为数组尾部，循环到数组尾部则跳出循环，否则跳转到第二个红圈中，判断当前数字是否为上一个数字的两倍，如果不是则爆炸。按照该要求可以猜出，这六个数字应该分别为1 2 4 8 16 32。



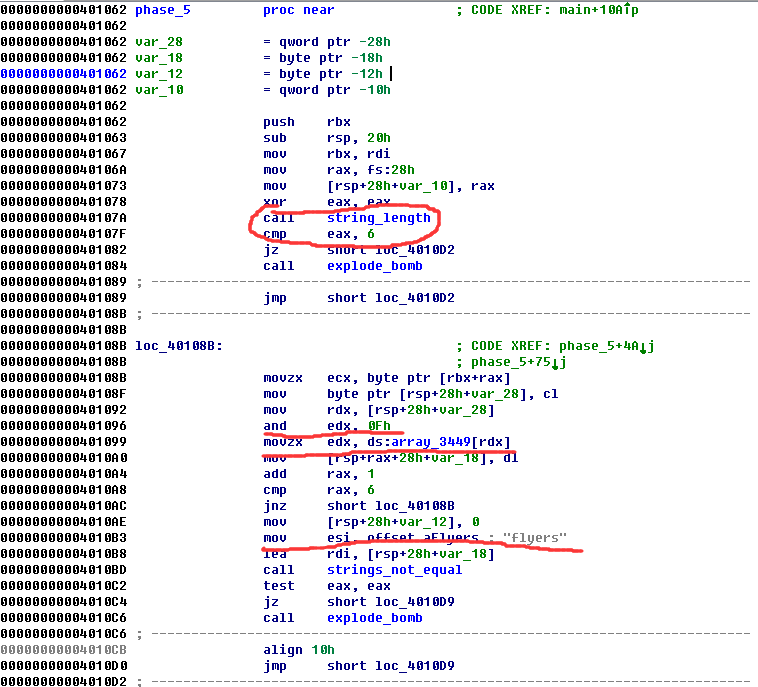
**Phase\_3**

第三阶段会将输入解析为两个数字，将第一个数字放入switch中，完成对eax的赋值，之后判断eax是否与输入的第二个数字相等，不难看出第一个数字为0的时候，eax的赋值为207，因此输入0 207即可完成第三阶段。



**Phase\_4**

第四阶段代码如上图所示，可以看到同样是需要输入两个数字，可以看到要求第二个输入数为0，且fun4(第一个数，0，14)函数的返回结果为0。进一步查看fun函数，可以发现当第一个数为7时函数会直接返回0，满足要求。因此输入应该为7 0。



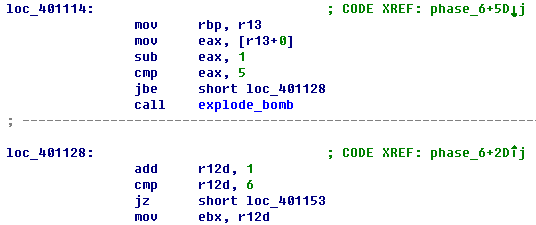
**Phase\_5**

第五阶段的代码如上图所示，要求输入的字符串长度为六，之后将每个字符与0xf进行按位与操作（可以理解为仅保留后四个有效二进制位），然后将结果放入字符组array\_3449中，要求得到的六个字符为”flyers”，查看array\_3449字符组如下：aduiersnfotvbyl，其中六个字符的对应的下标分别为9 15 14 5 6 7，二进制位为1001 1111 1110 0101 0110 0111，找到后四位分别为上述六个的ASCII码即可，例如)/.%&’。

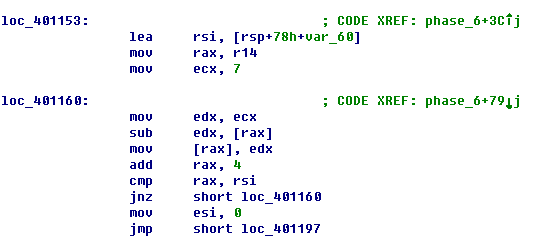


**Phase\_6**

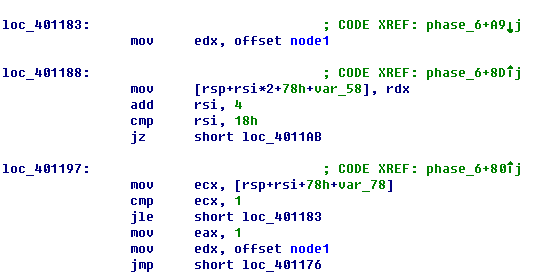
第六阶段如上图所示，同样需要输入六个字符，之后汇编过于繁杂，只记录关键部分的分析。



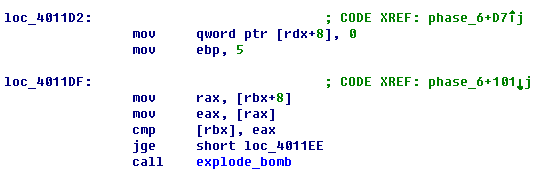
如上图所示，该部分对输入的六个数字进行检查，如果有一个数字减一大于五则爆炸，因此输入的每个数字都应该小于等于6。



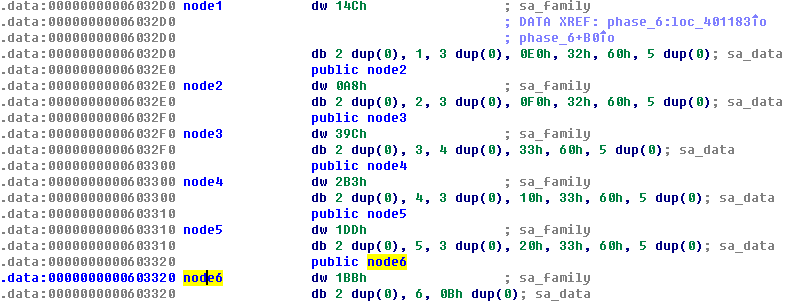
如上图所示，程序会用7减去输入的每个数字并进行覆盖代替输入的六个数字。



如上图所示，该部分代码根据六个对应的数字取出六个node，node共有六个，每一个是一个结构体，每一个结构体中都有一个值并包含下一个node的地址。



该部分代码要求取出的六个Node按从大到小的顺序排列，否则爆炸。查看六个Node的值，其中的内容如下所示：



从大到小排列为3 4 5 6 1 2，用7减过后得4 3 2 1 6 5，则正确的输入应该是4 3 2 1 6 5。

综合六个阶段的分析，防止爆炸输入如下所示：

Border relations with Canada have never been better.

1 2 4 8 16 32

0 207

7 0

)/.%&'

4 3 2 1 6 5