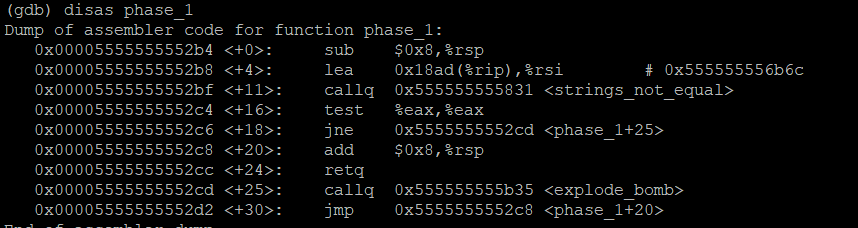
**Bomb Lab**

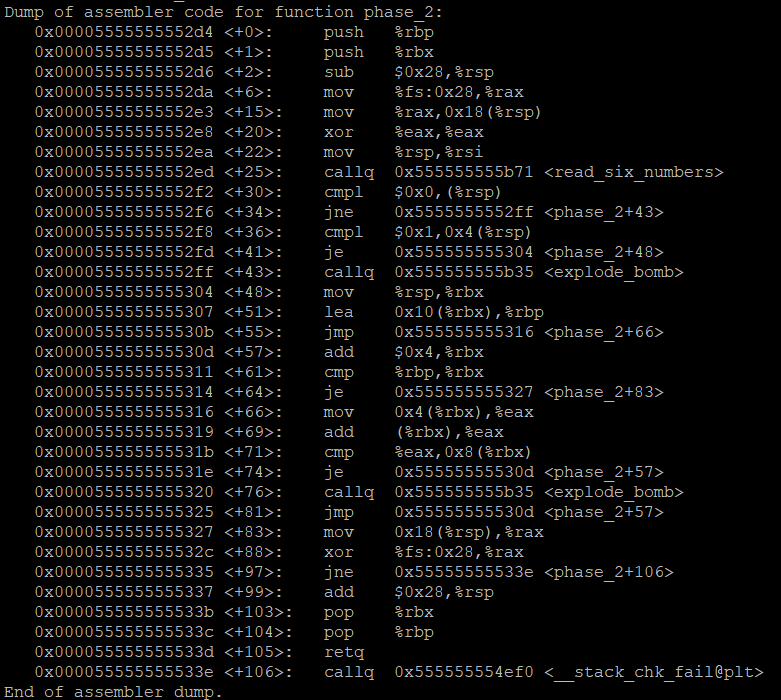
\*기본적으로 explode\_bomb에 breakpoint를 걸고 작업했습니다.

**Phase\_1**



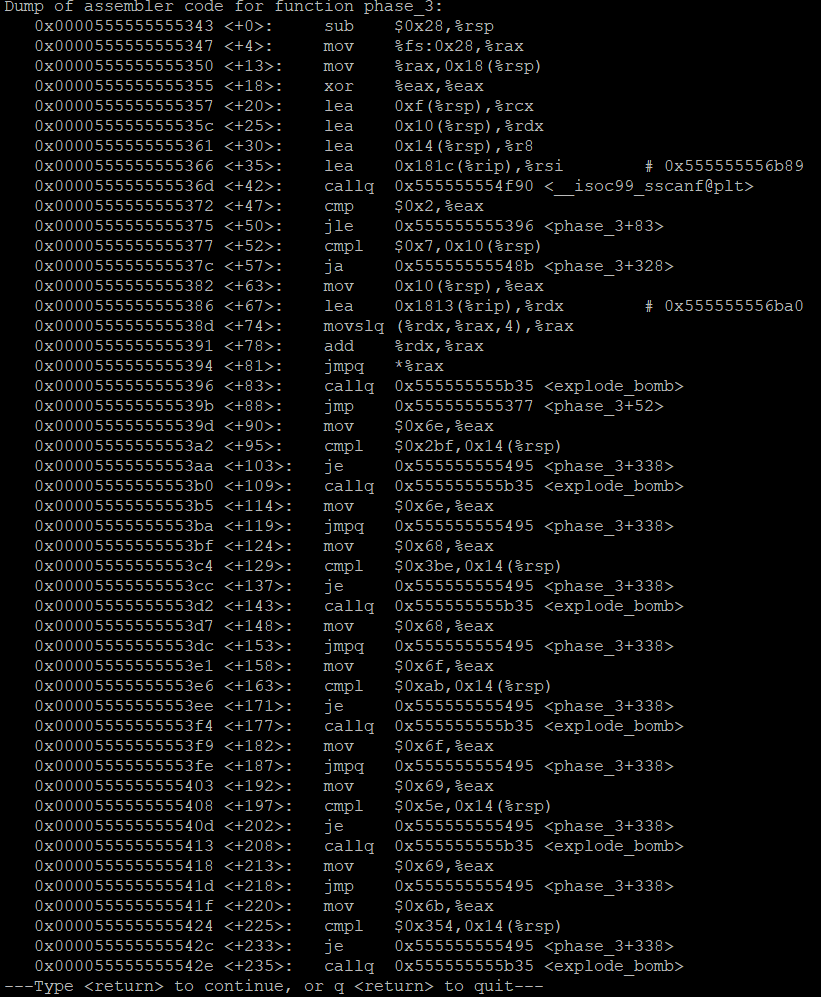
<String\_not\_equal>이 있으므로 입력 받아야 할 것은 string이라는것을 알 수 있다.<String\_not\_equal>이 실행되기 직전의 레지스터, 즉 %rsi를 확인하면 “Crikey! I have lost my mojo!”라는 문자열을 발견할 수 있다. 이 문자열이 phase\_1을 통과하는 열쇠가 된다.

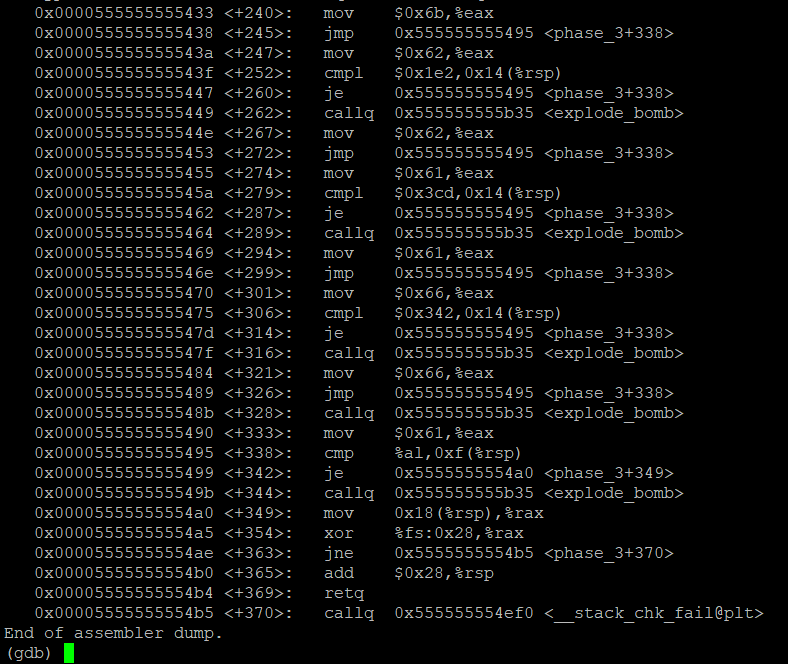
**Phase\_2**



<read\_six\_number>가 있는것을 보면 입력할 것이 6개의 숫자라는것을 알 수 있다.그리고 그 다음 0을 비교하고 그 다음 1을 비교한다. 즉 처음 두개의 숫자는 0과 1이라는것을 알 수 있다. 그리고 중간에 보면 loop가 계속되는것을 확인할 수 있는데, %rbp와 %rbx 그리고 %eax를 잘 보다 보면 이전값+지금값 = 다음값 이라는 사실을 알 수 있다. 따라서 답은 0 1 1 2 3 5 가 된다.

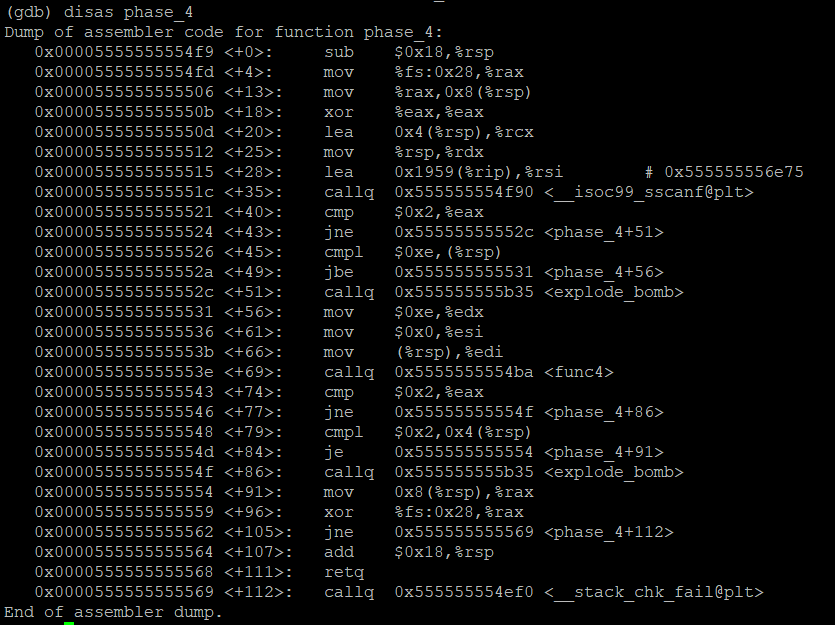
**Phase\_3**



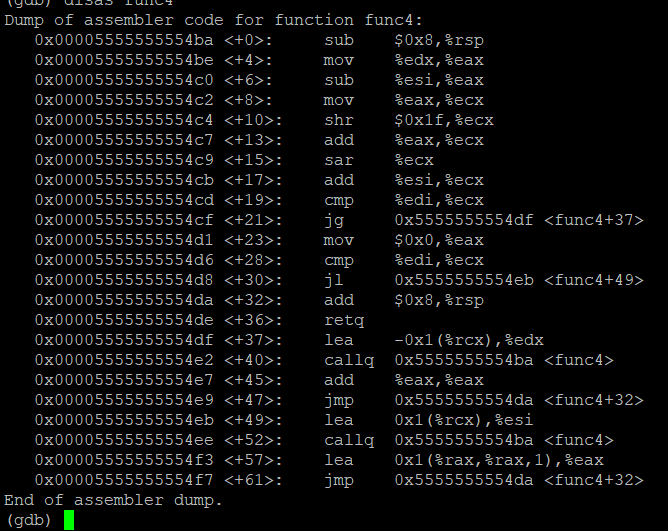


초반에 scanf가 불리는것을 확인 할 수 있는데, x/s 명령으로 직전에 주어지는 0x555555556b89를 확인해 보면 “%d %c %d”가 뜬다. 즉 여기서 입력은 숫자, 문자, 숫자 라는 뜻이 된다. 그리고 0x7과 cmpl을 하는것을 보아 첫번째 숫자는 7보다 작아야 한다는 뜻이 된다. 그리고 두번째 이루어지는 작업은 +328부터 이루어지는데, %al레지스터와 관련이 있었다. 여러 숫자들을 넣어가며 확인해 본 결과 %al레지스터의 값이 입력값에 따라 바뀐다는 것을 알게 되었고, 두번째 입력은 문자이니 아스키 코드라고 추측을 하였다. 3을 넣으니 %al에 105라는 수가 들어있었다. 즉 3일때는 i를 입력해야 한다고 추측할 수 있다. 그 다음은 입력한 것에 따라 계산한 주소에 따라서 점프를 하게 되는데, 그 주소에 따라 입력되는 %eax를 확인해 보니 94가 나왔다. 따라서 답은 “3 i 94”가 된다.

**Phase\_4**

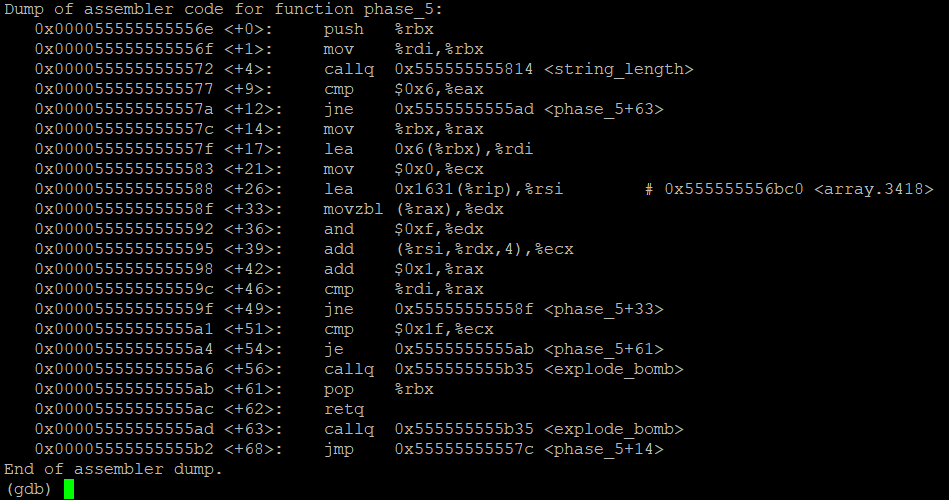


이번에도 scanf입력을 받기 전에 x/s로 0x555555556e75를 확인해보면 “%d %d”가 나온다. 즉 입력할 것은 숫자 2개가 된다. 일단 cmpl 0xe(14) 를 보면 첫 입력은 14보다 작아야 함을 알 수 있다. 그리고 <func4>를 호출 함을 알 수 있다. 그리고 2가 되는지 확인하여 다음 단계로 가는 것을 볼 수 있다.

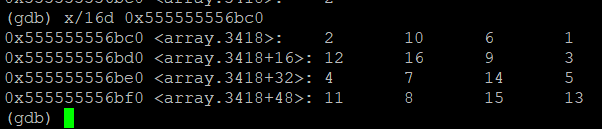


Func4를 확인한 결과 5를 넣으면 2가 결과로 나오는 걸 확인할 수 있다. 그리고 +79에서 0x2와 (%rsp)0x4를 비교하므로 두번째 값은 2라는것을 확인 할 수 있다. 즉 답은 “5 2”가 된다.

**Phase\_5**



<string\_length> 밑의 0x6과 비교하는 cmp명령을 보니 6개의 문자로 구성된 문자열을 입력해야 함을 알 수 있다. 그리고 시행착오 끝에 확인한 x/16d명령으로 0x555555556bc0를 보면 숫자의 배열이 있는것을 확인 할 수 있었다.



첫 숫자는 공백이고 a는 10, b는 6... 이런식으로 문자와 숫자가 연결됨을 알 수 있다. 그리고 그 맵핑된 수들을 더해서 +51에서 0x1f와 비교를 한다. 여러가지 경우가 있겠지만, 여기서는 “bcfghm”(6+1+9+3+4+8=31)로 답을 정하였다.