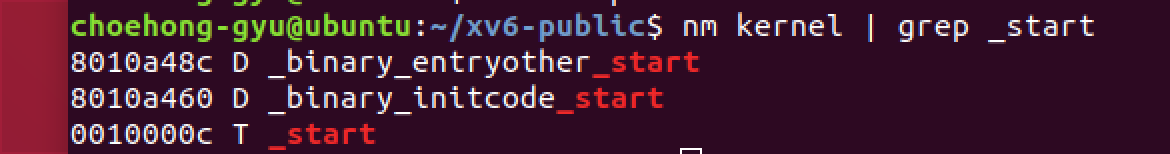
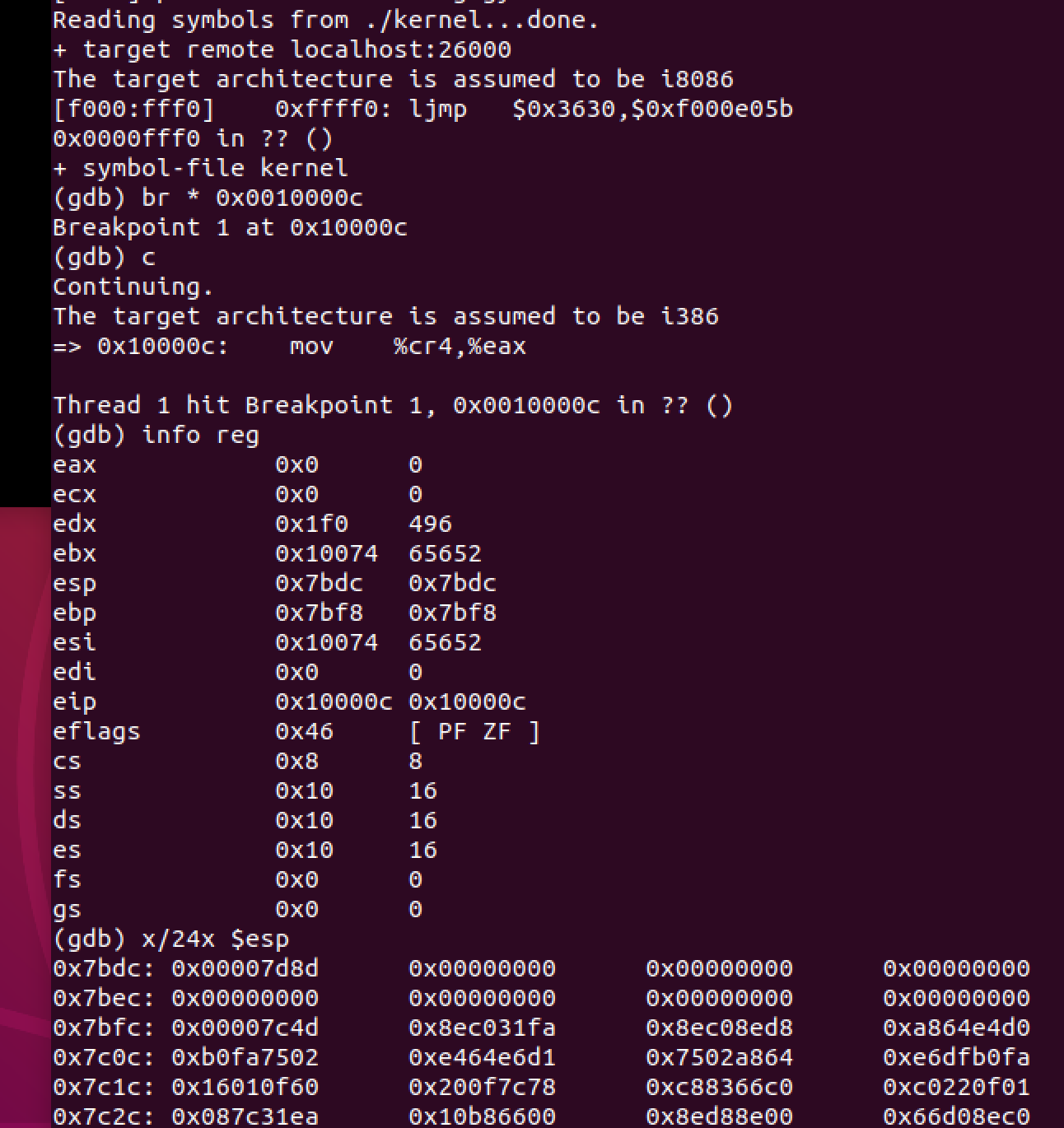
**Homework1 : xv6 부팅하기**

**2015005205 최홍규**



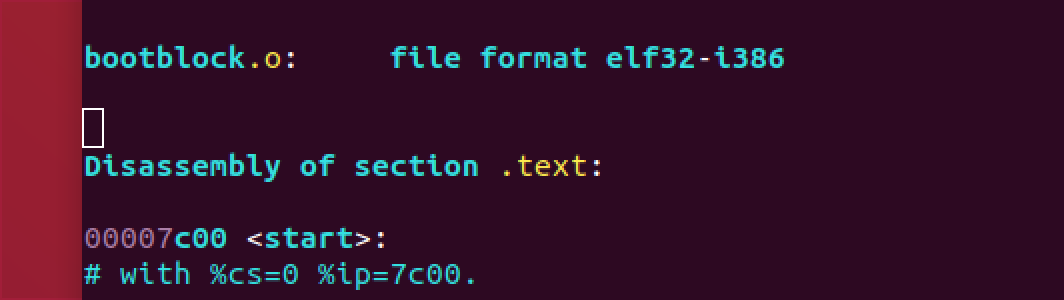
kernel entry point의 주소를 찾아보았다. 주소값는 0x0010000c 였다.

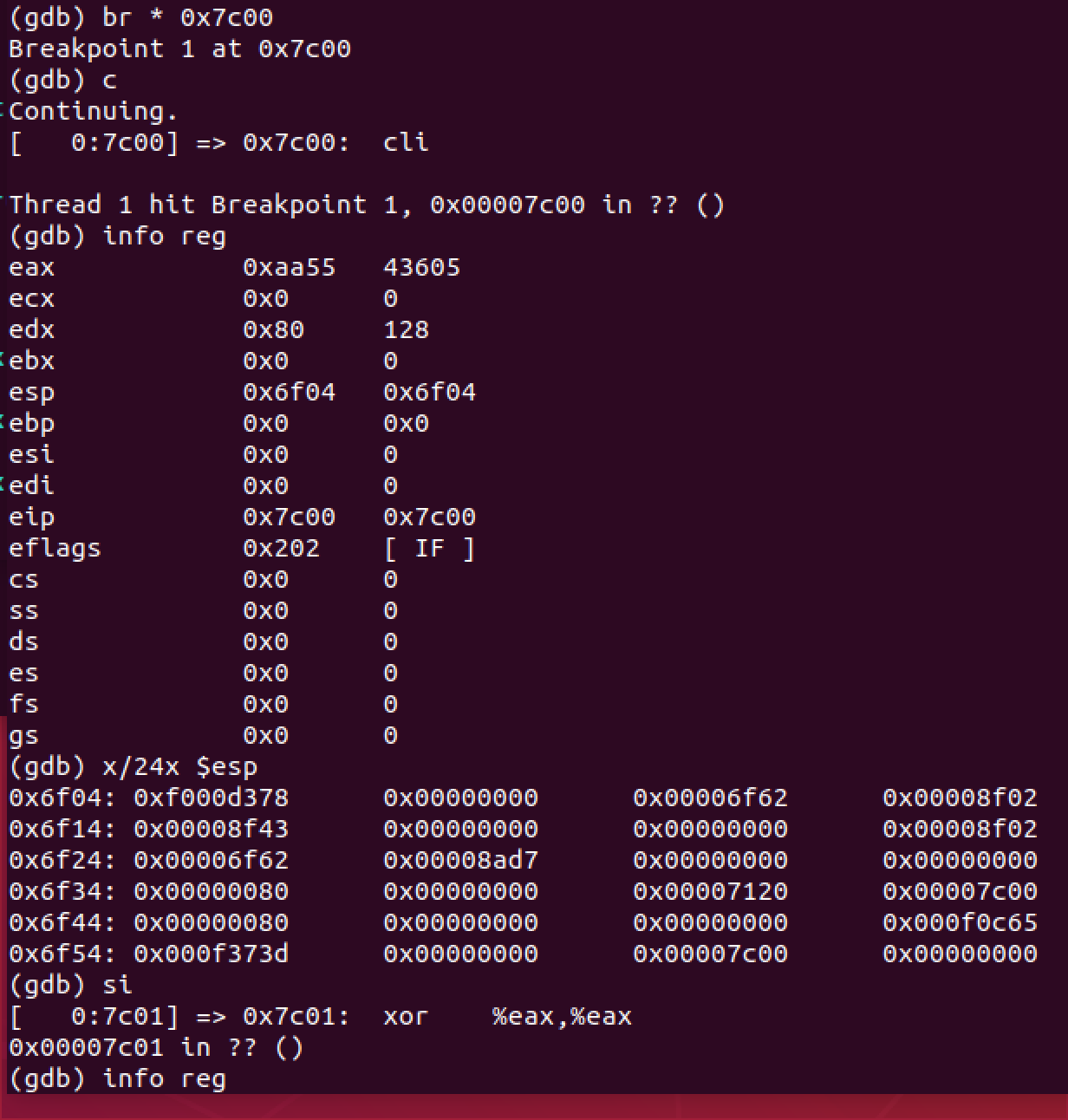


kernel entry point까지 코드를 실행했을 때 register들의 값과 stack의 내용들을 확인하였다.

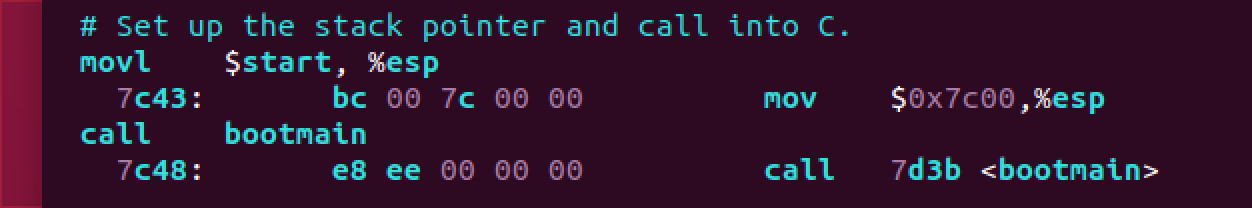
현재 보고있는 코드의 위치인 eip 값은 breakpoint를 걸어둔 0x0010000c이고 stack의 가장 위의 주소를 나타내는 esp 값은 0x00007bdc, stack의 가장 바닥을 나타내는 ebp 값은 0x00007bf8이다. 이때의 실제 stack은 어디이고 0이 아닌 값들은 무엇을 의미하는지 알아보기 위해 booting이 시작 되는 주소를 찾아보고 register값들의 변화를 보며 stack의 변화를 tracking 해보자.

1. **bootasm.S의 어느 부분에서 stack pointer가 초기화 되는가?**

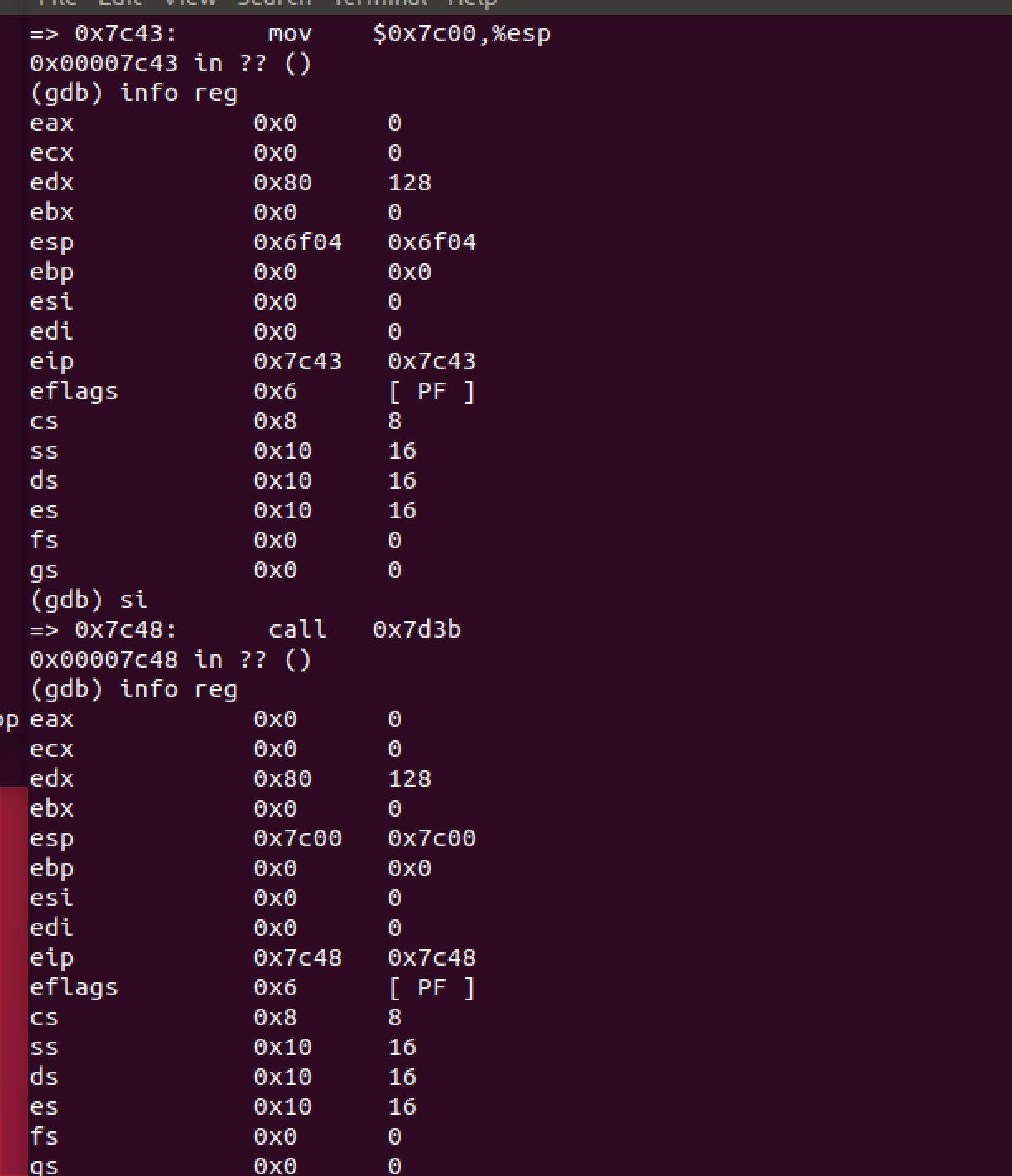




bootblock의 시작 위치는 0x00007c00인 것을 확인하였다. bootblock의 시작점인 0x00007c00에 break point를 지정하고 탐색을 시작해 보았다.

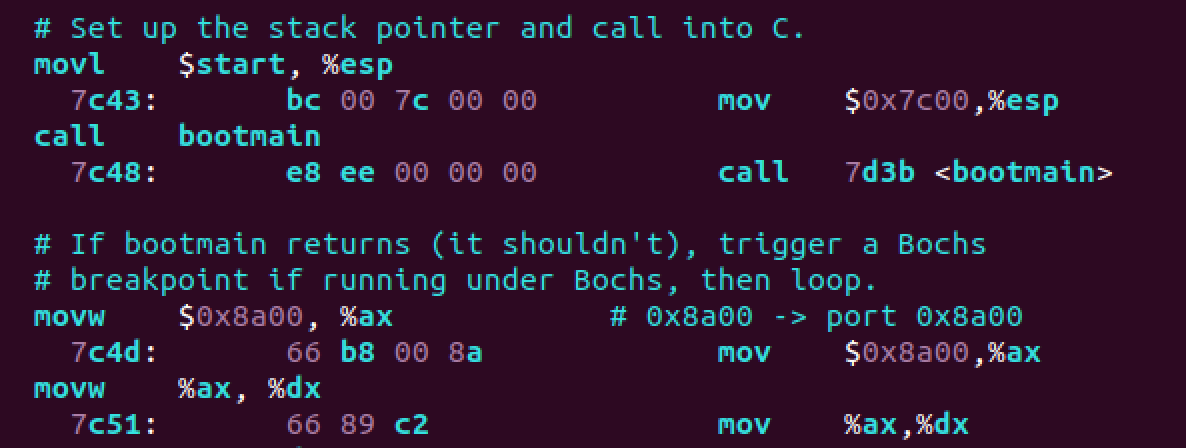


bootblock.asm 파일에 있는 코드를 살펴보던중 bootmain으로 들어가기 전 esp(stack pointer)값을 0x00007c00으로 초기화해주는 코드를 발견하였다. break point를 0x00007c43에 두고 진행해보았다.

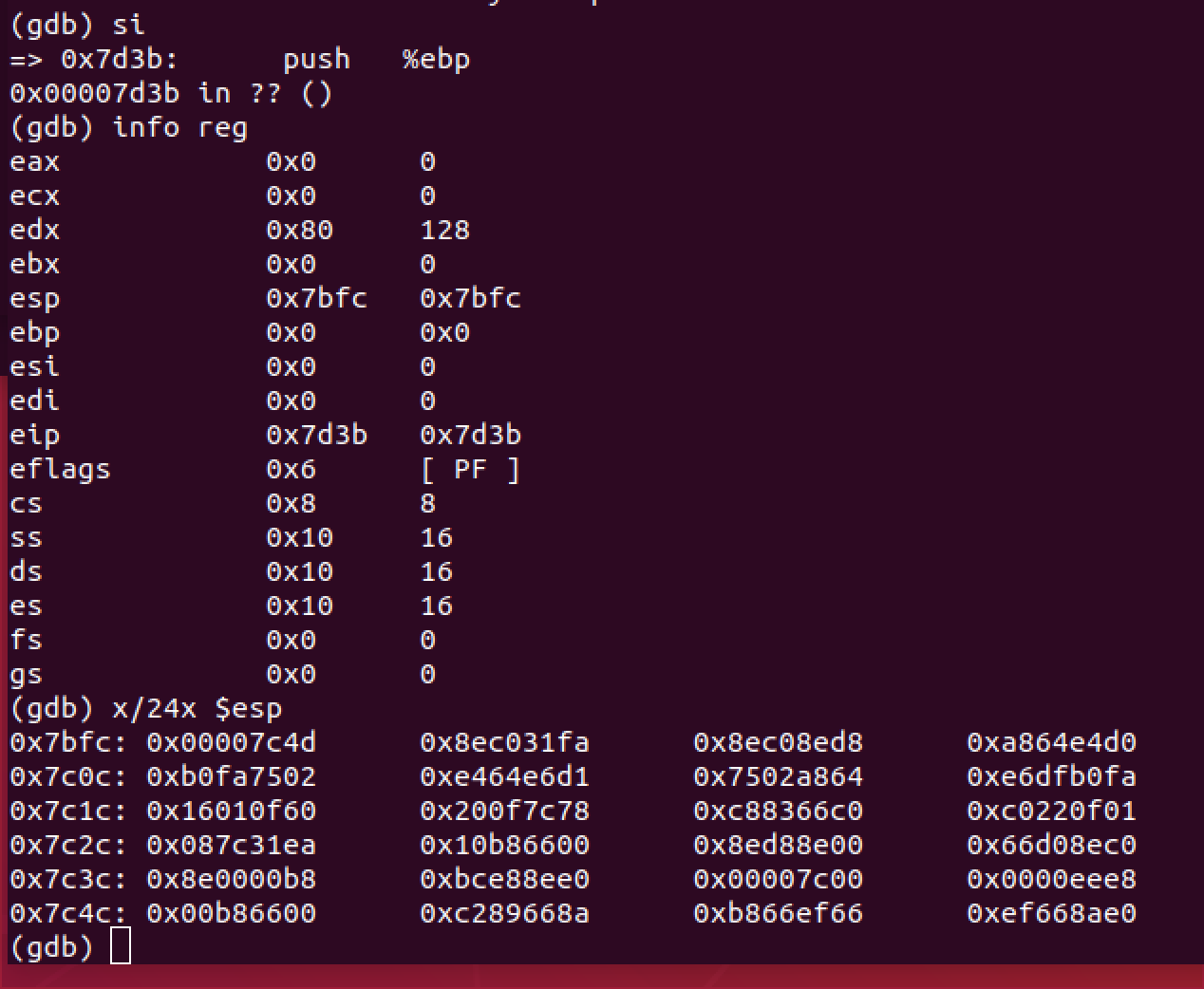


esp(stack pointer)가 0x00006f04에서 0x00007c00으로 초기화 되었다.

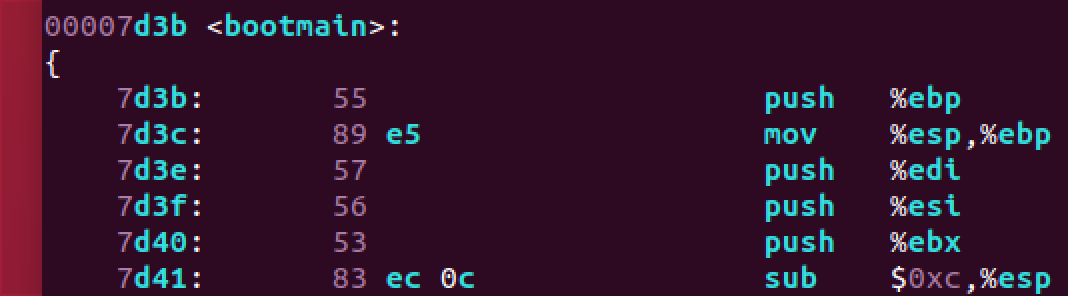
1. **bootmain의 코드를 한단계씩 수행한다. 현재 stack에는 무엇이 있는가?**



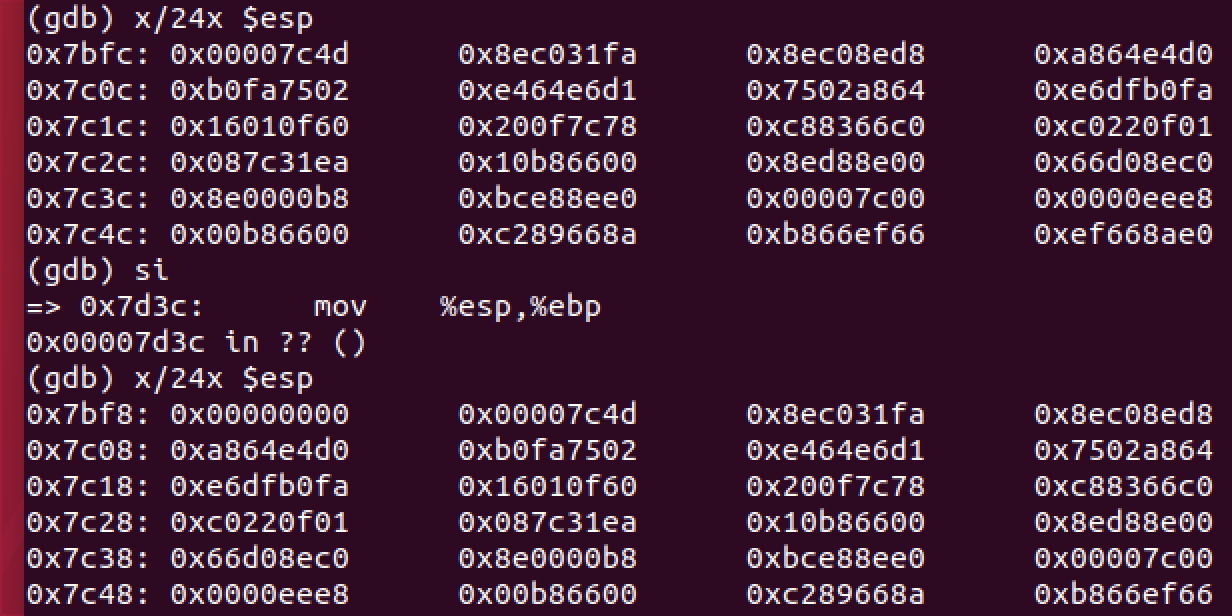
bootmain이 끝난 뒤 return되는 주소값은 0x00007c4d이다. bootmain을 call하는 작업을 할 때 0x00007c4d 값이 stack안에 저장되는 것 밑의 그림에서 확인 할 수 있다. 따라서 bootmain의 return address가 stack에 저장됨을 볼 수 있다.

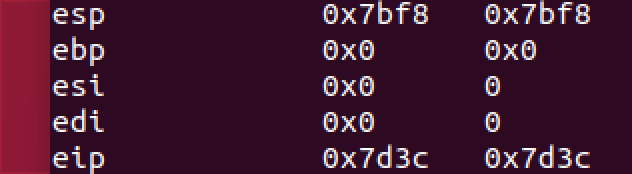
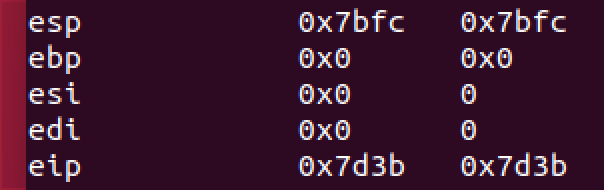


1. **bootmain의 첫 번째 assembly 명령어들이 stack에서 하는 일은 무엇인가?**



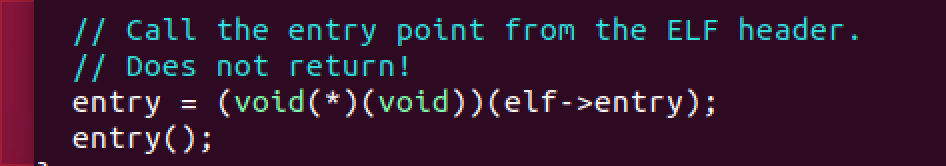
bootmain의 첫줄을 실행하였다.

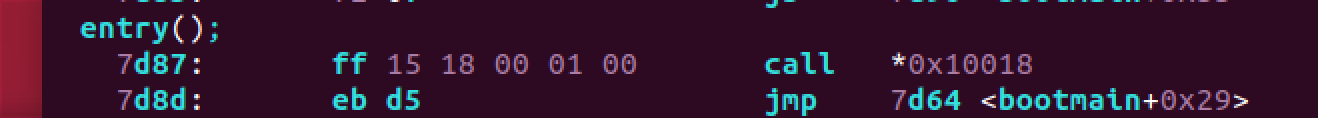


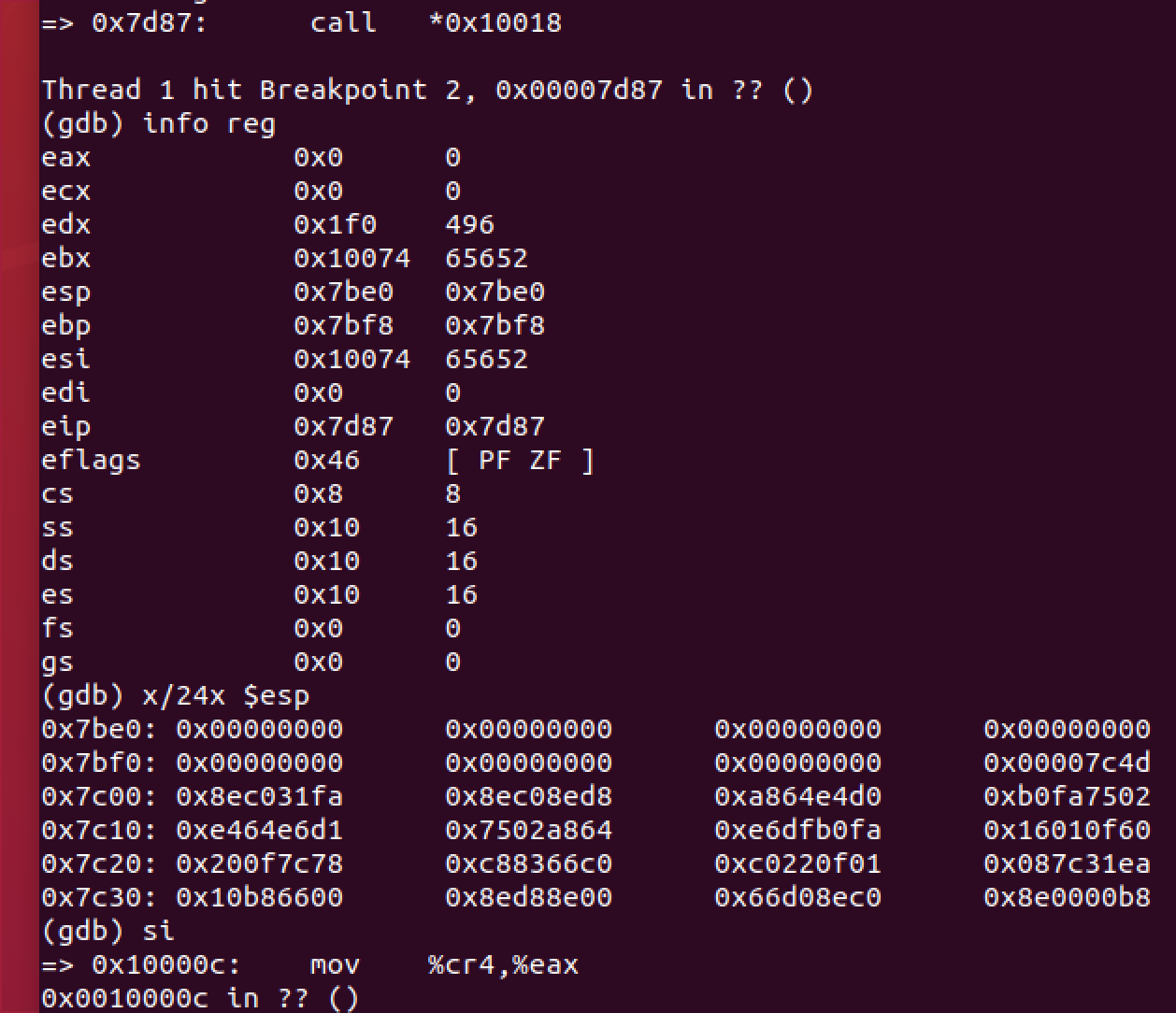


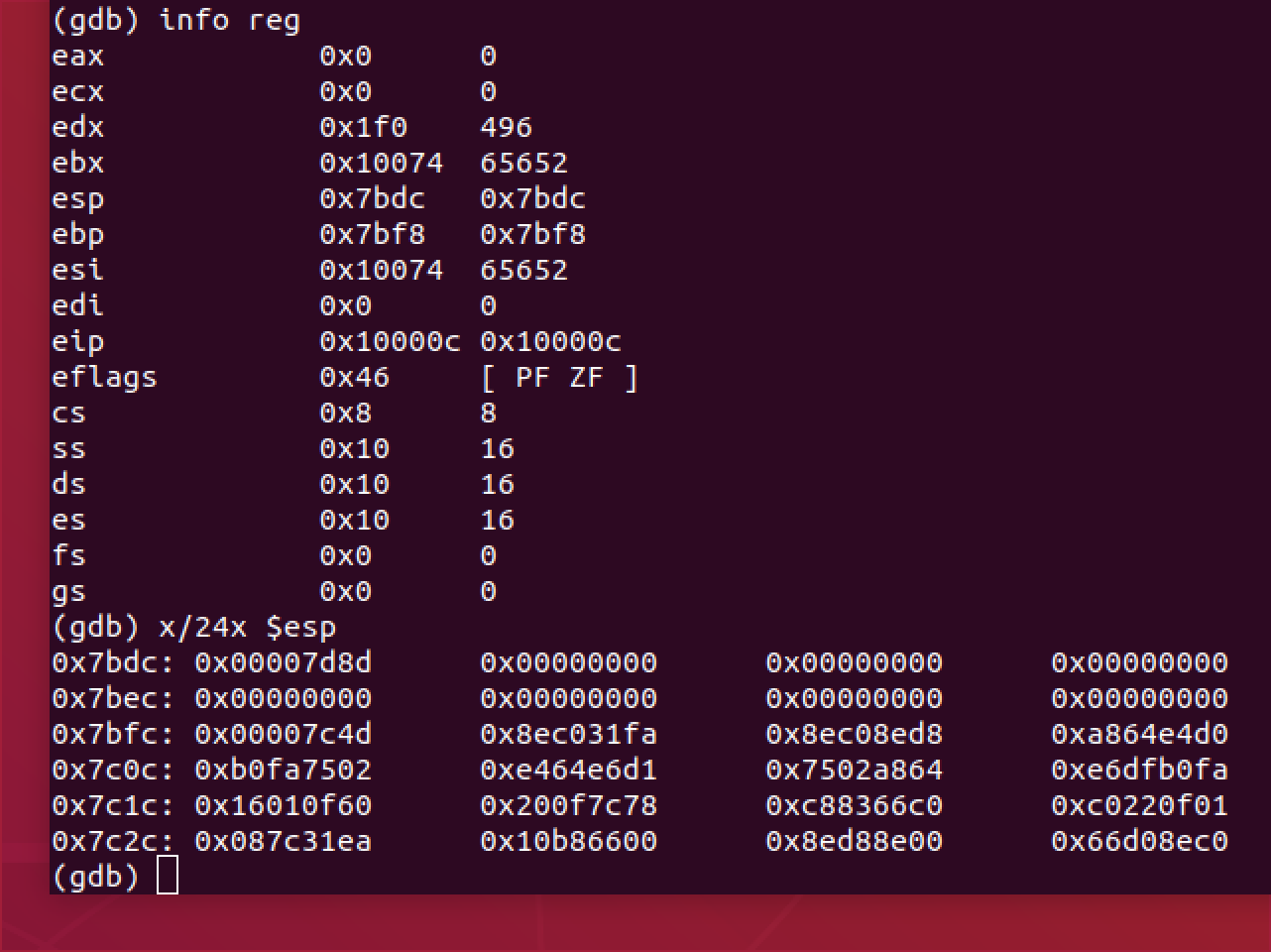
push %ebp에 의해 stack에 ebp값인 0x0을 저장하였다. stack의 top을 나타내는 esp값이 4감소했다. stack의 시작값을 저장하는 ebp값에 esp값을 넣어줌으로써 bootmain에 속하는 stack을 구분해주었다. 그 이후 여러 register의 값을 stack에 저장해주는 과정이 있다.

1. **gdb를 통해 eip를 0x10000c로 변경하는 코드를 탐색한다. 그 코드가 stack에서 하는 일은 무엇이가?**



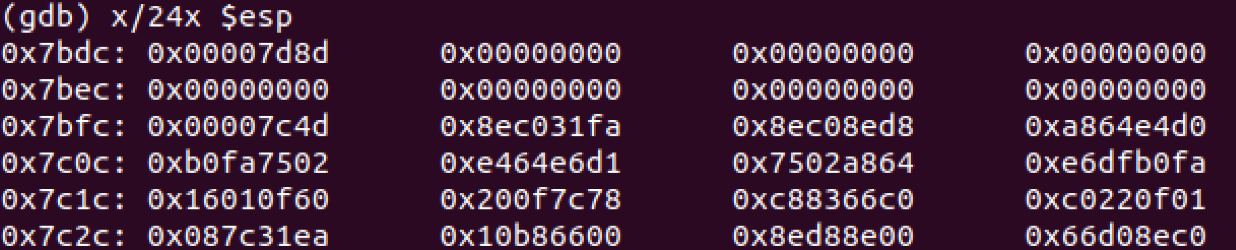






booting을 하며 결국 kernel로 들어가는게 bootmain의 목적이다. 따라서 boot sequence를 완료하기 위해서 kernel로 들어가야한다. entry함수를 통해 call \*0x10018 코드를 실행한 후 eip값이 0x10000c로 변했다. 이 값은 \_start(kernel entry point) 주소값과 같은 값이다. 따라서 entry함수의 call \*0x10018은 kernel로 들어가는 코드 을 알 수 있었다. stack에는 entry함수의 return address를 저장해 주었다.

\*stack에서 0이 아닌 값들이 무엇을 의미하는 지 3~5단어로 기술하라. 출력문 의 어느 부분이 실제 stack인가?



처음으로 돌아가 kernel로 들어가기 전 stack의 상태이다. booting 후 사용하는 실제 stack 영역은bootmain return address를 사용하는 부분부터이다. 따라서 실제 stack 부분은 bootmain의 return address를 저장하는 위치부터 이므로 빨간색 박스로 체크한 부분 0x00007bdc ~ 0x00007bff까지 인 것을 알 수 있다. stack 내부의 파란색 박스의 값들은 xv6를 booting할 때 호출하는 bootmain(0x00007c4d)과 entry함수(0x00007d8d)의 return address값을 나타낸다. 따라서 stack에서 0이 아닌 값들이 의미하는 것은 “부팅시 호출되는 함수의 return address”이다.