Lab 11: Binary Analysis and Exploitation

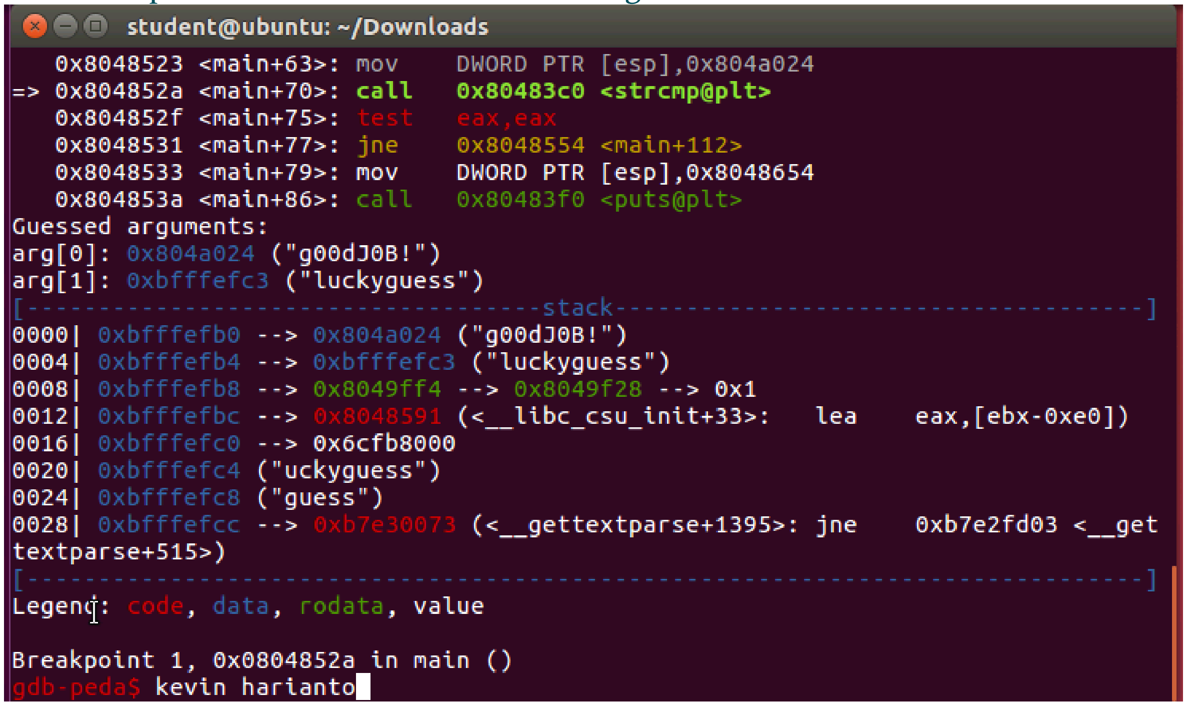
INFO40587: ETHICAL HACKING

Kevin Harianto| 991602128| July 30, 2024

Exercise 1: Binary Analysis

1.1 OUTPUT SCREENSHOTS

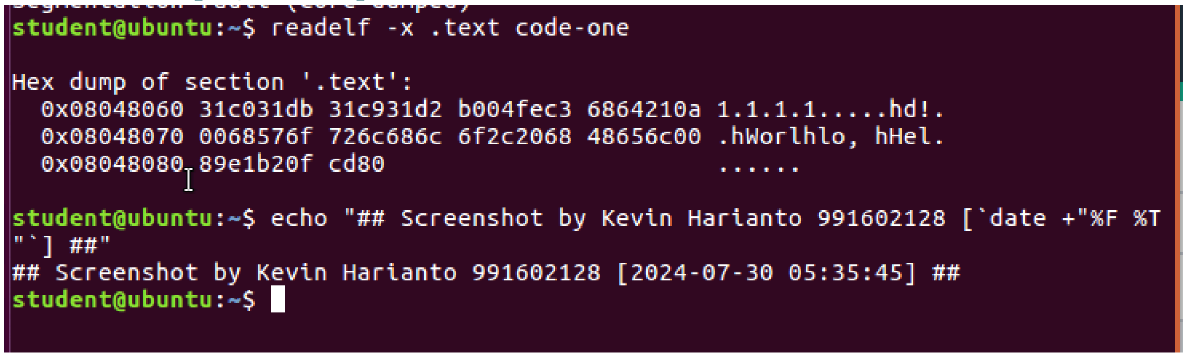
Exercise 1, Step 18: Enter the password **luckyguess**. The comparison will reference the actual password as shown in the following screenshot



```
student@ubuntu: ~/Downloads
   0x8048523 <main+63>: mov    DWORD PTR [esp],0x804a024
=> 0x804852a <main+70>: call   0x80483c0 <strcmp@plt>
   0x804852f <main+75>: test   eax,eax
   0x8048531 <main+77>: jne    0x8048554 <main+112>
   0x8048533 <main+79>: mov    DWORD PTR [esp],0x8048654
   0x804853a <main+86>: call   0x80483f0 <puts@plt>
Guessed arguments:
arg[0]: 0x804a024 ("g00dJ0B!")
arg[1]: 0xbfffefc3 ("luckyguess")
[-----------------------------------stack------------------------------------]
0000| 0xbfffefb0 --> 0x804a024 ("g00dJ0B!")
0004| 0xbfffefb4 --> 0xbfffefc3 ("luckyguess")
0008| 0xbfffefb8 --> 0x8049ff4 --> 0x8049f28 --> 0x1
0012| 0xbfffefbc --> 0x8048591 (<__libc_csu_init+33>:   lea    eax,[ebx-0xe0])
0016| 0xbfffefc0 --> 0x6cfb8000
0020| 0xbfffefc4 ("uckyguess")
0024| 0xbfffefc8 ("guess")
0028| 0xbfffefcc --> 0xb7e30073 (<__gettextparse+1395>: jne    0xb7e2fd03 <__get
textparse+515>)
[----------------------------------------------------------------------------]
Legend: code, data, rodata, value

Breakpoint 1, 0x0804852a in main ()
gdb-peda$ kevin harianto
```

Exercise 1, Step 65: Lookup result of 'certifiedhacker.com' domain

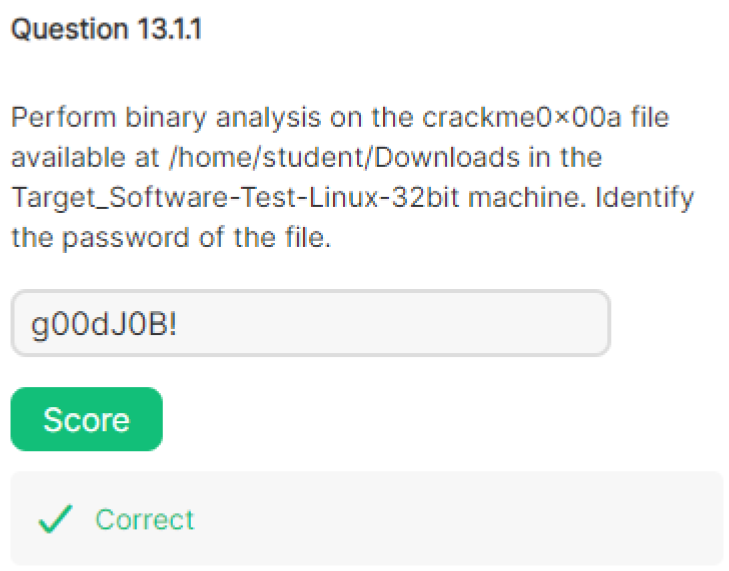


```
student@ubuntu:~$ readelf -x .text code-one

Hex dump of section '.text':
  0x08048060 31c031db 31c931d2 b004fec3 6864210a 1.1.1.1.....hd!.
  0x08048070 0068576f 726c686c 6f2c2068 48656c00 .hWorlhlo, hHel.
  0x08048080 89e1b20f cd80                       ......

student@ubuntu:~$ echo "## Screenshot by Kevin Harianto 991602128 [`date +"%F %T
"`] ##"
## Screenshot by Kevin Harianto 991602128 [2024-07-30 05:35:45] ##
student@ubuntu:~$ 
```

1.2 Questions



Question 13.1.1

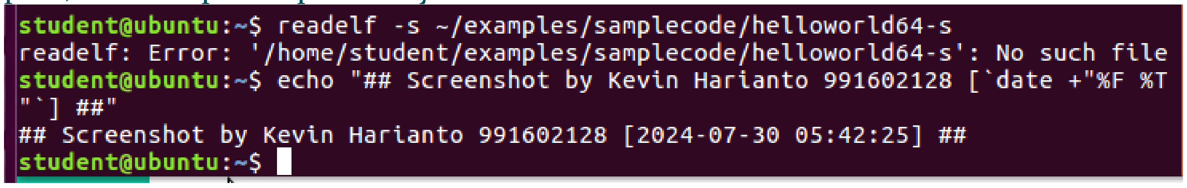
Perform binary analysis on the crackme0×00a file available at /home/student/Downloads in the Target_Software-Test-Linux-32bit machine. Identify the password of the file.

g00dJOB!

Exercise 2: Binary Analysis on a 64-bit Machine

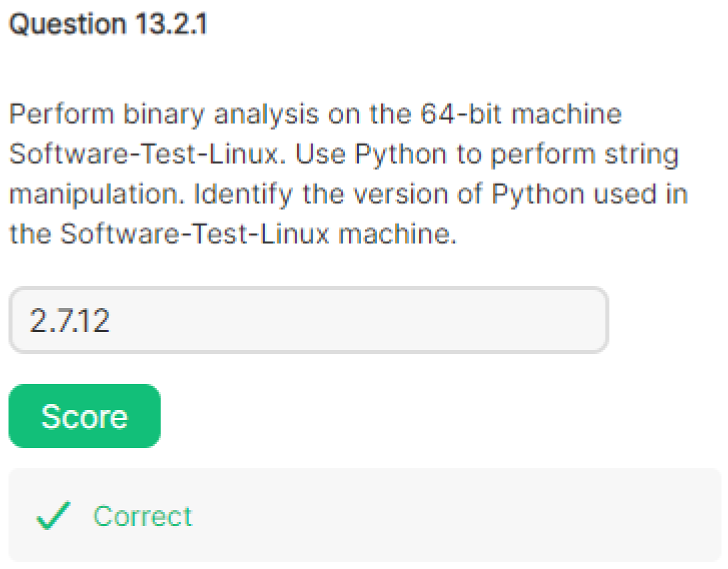
2.1 OUTPUT SCREENSHOTS

Exercise 2, Step 12: Next, enter **readelf -s ~/examples/samplecode/helloworld64-s**. The output of the symbol table is shown in the following screenshot.
NOTE: Despite executing the python commands previously and using the same file path, the example file previously made was unable to be found.



```
student@ubuntu:~$ readelf -s ~/examples/samplecode/helloworld64-s
readelf: Error: '/home/student/examples/samplecode/helloworld64-s': No such file
student@ubuntu:~$ echo "## Screenshot by Kevin Harianto 991602128 [`date +"%F %T
"`] ##"
## Screenshot by Kevin Harianto 991602128 [2024-07-30 05:42:25] ##
student@ubuntu:~$ 
```

2.2 QUESTIONS


Question 13.2.1

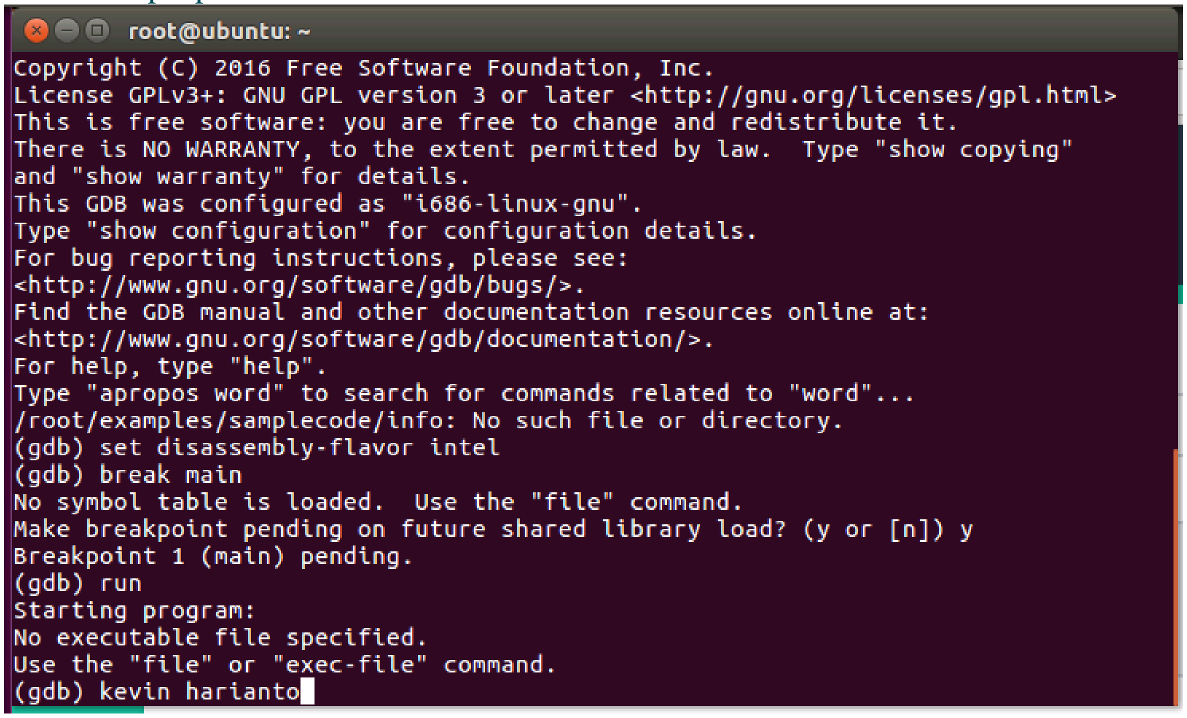
Perform binary analysis on the 64-bit machine Software-Test-Linux. Use Python to perform string manipulation. Identify the version of Python used in the Software-Test-Linux machine.

2.7.12

Exercise 3: Binary Analysis Methodology

3.1 OUTPUT SCREENSHOTS

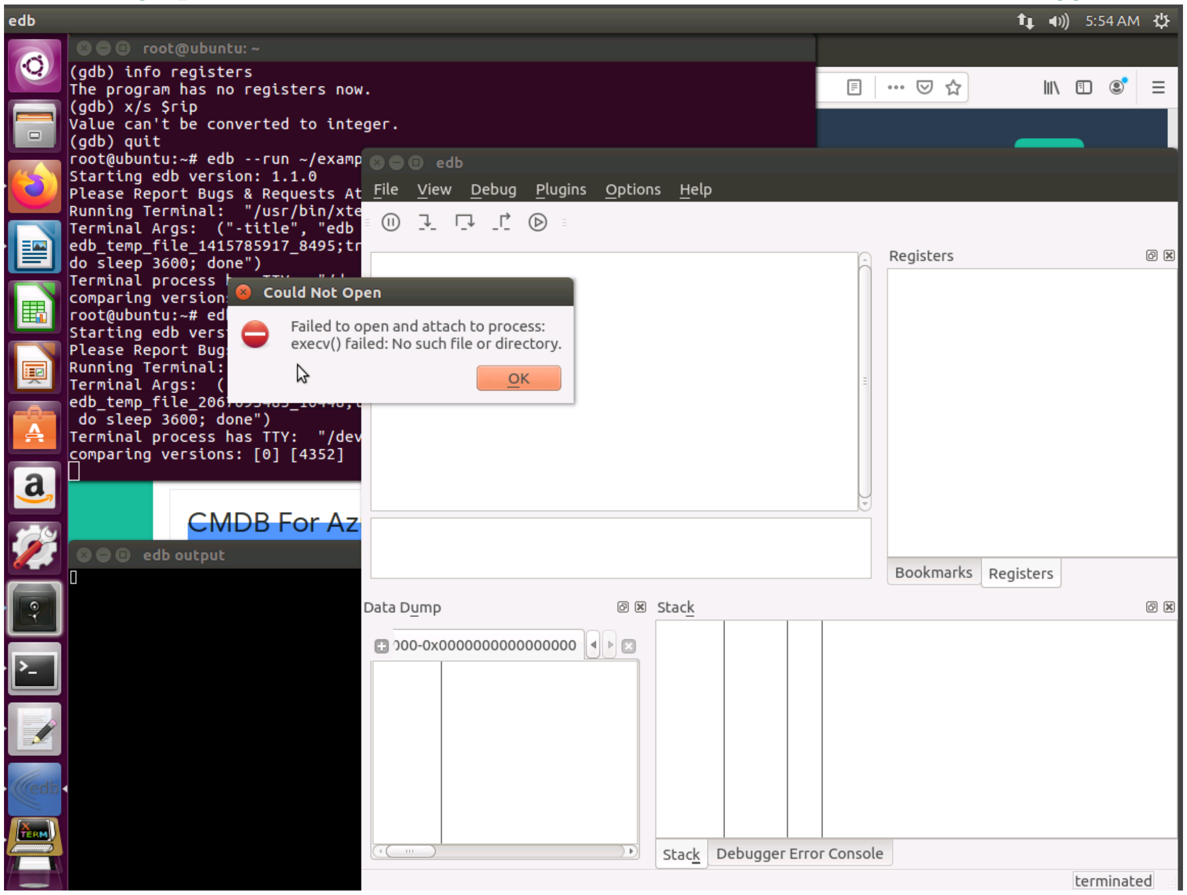
Exercise 3, Step 32: An example of when the program hits the breakpoint is shown in the following screenshot.
NOTE: Despite following the GDB commands I was unable to load the symbol table. Nevertheless, I was still able to get the breakpoint and debugger running for education purposes.



```
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "i686-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
/root/examples/samplecode/info: No such file or directory.
(gdb) set disassembly-flavor intel
(gdb) break main
No symbol table is loaded.  Use the "file" command.
Make breakpoint pending on future shared library load? (y or [n]) y
Breakpoint 1 (main) pending.
(gdb) run
Starting program:
No executable file specified.
Use the "file" or "exec-file" command.
(gdb) kevin harianto
```

Exercise 3, Step 39: click on the **Find** button, and look for the info main function in the symbol table. Click on it, and then select **Graph Selected Function**. Take a few minutes to review the results from the graphing of the function.

NOTE: Due to being unable to open the debugger to analyze the file I was unable to enter the graph section. Nevertheless, I was still able to launch the evan debugger.



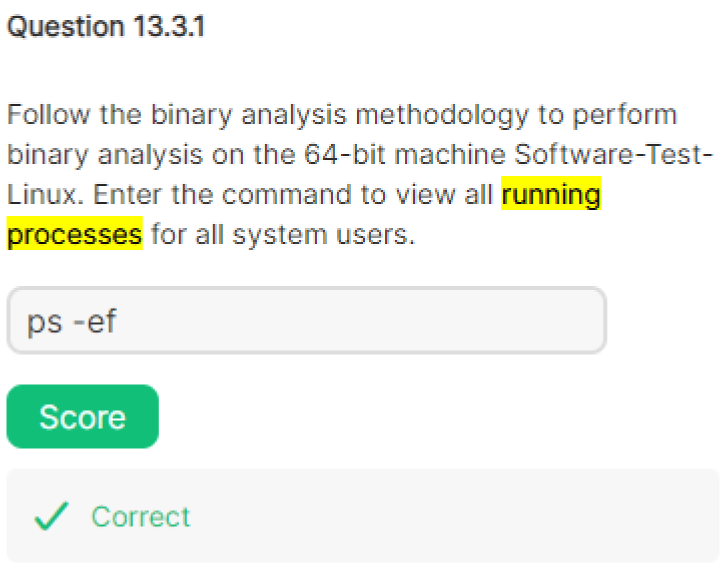
```
(gdb) info registers
The program has no registers now.
(gdb) x/s $rip
Value can't be converted to integer.
(gdb) quit
root@ubuntu:~# edb --run ~/examp
Starting edb version: 1.1.0
Please Report Bugs & Requests At
Running Terminal:  "/usr/bin/xte
Terminal Args:  ("-title", "edb
edb_temp_file_1415785917_8495;tr
do sleep 3600; done")
Terminal process h
comparing version
root@ubuntu:~# ed
Starting edb vers
Please Report Bug
Running Terminal:
Terminal Args:  (
edb_temp_file_206
 do sleep 3600; done")
Terminal process has TTY:  "/dev
comparing versions: [0] [4352]
```

Could Not Open

Failed to open and attach to process:
execv() failed: No such file or directory.

OK

3.2 Questions


Question 13.3.1

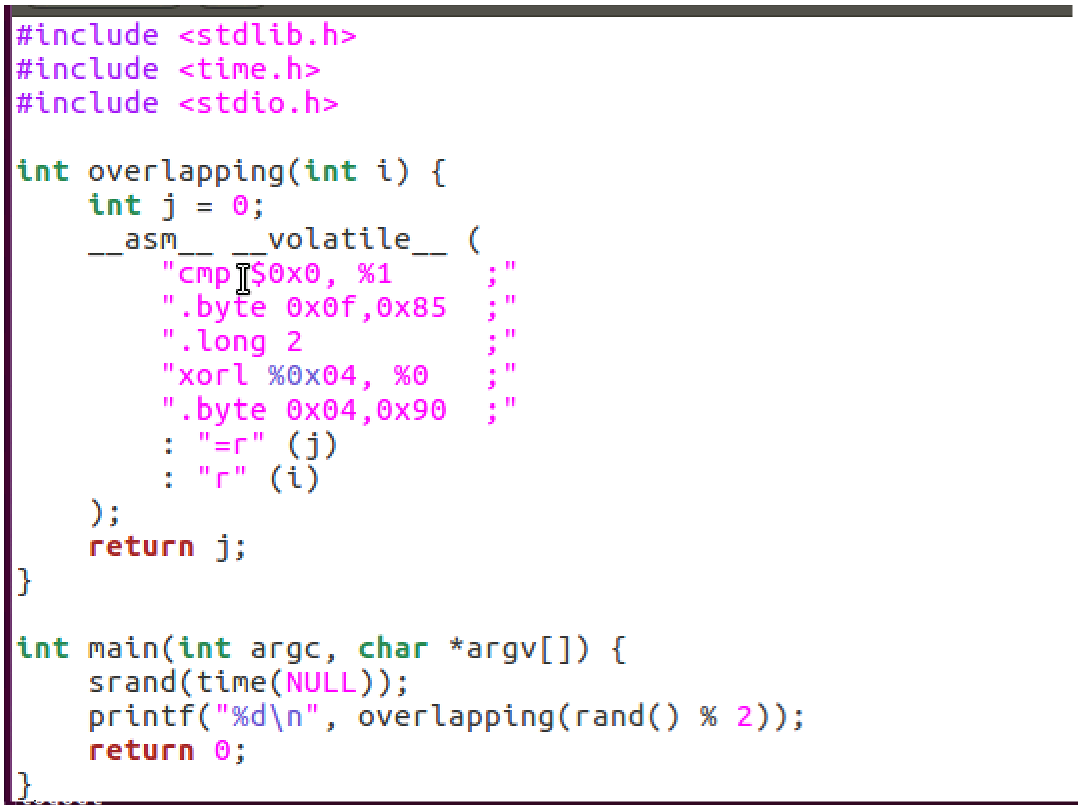
Follow the binary analysis methodology to perform binary analysis on the 64-bit machine Software-Test-Linux. Enter the command to view all running processes for all system users.

```
ps -ef
```

Exercise 4: Advanced Binary Analysis

4.1 OUTPUT SCREENSHOTS

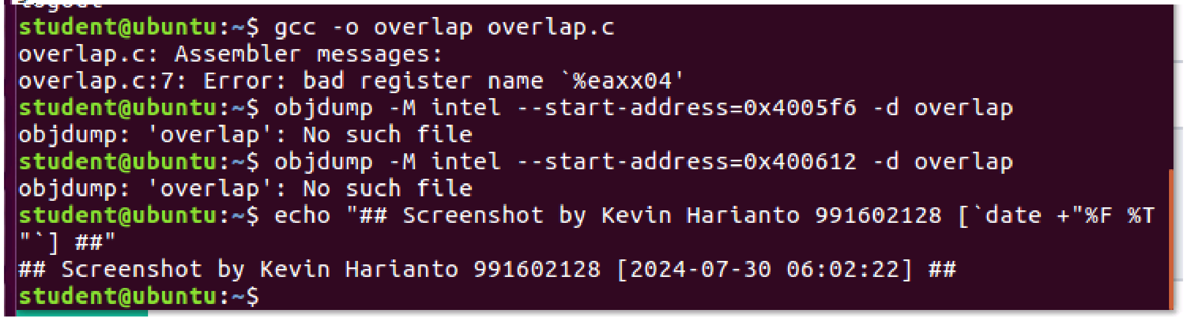
Exercise 4, Step 16: enter **objdump -M intel --start-address=0x400612 -d overlap**. An example of the output of this is shown in the following screenshot.
NOTE: despite typing in the entire code and following the gcc commands instructed. I was unable to load the register name due to a possible update in the OS.



```c
#include <stdlib.h>
#include <time.h>
#include <stdio.h>

int overlapping(int i) {
    int j = 0;
    __asm__ __volatile__ (
        "cmp $0x0, %1      ;"
        ".byte 0x0f,0x85   ;"
        ".long 2           ;"
        "xorl %0x04, %0    ;"
        ".byte 0x04,0x90   ;"
        : "=r" (j)
        : "r" (i)
    );
    return j;
}

int main(int argc, char *argv[]) {
    srand(time(NULL));
    printf("%d\n", overlapping(rand() % 2));
    return 0;
}
```



```
student@ubuntu:~$ gcc -o overlap overlap.c
overlap.c: Assembler messages:
overlap.c:7: Error: bad register name `%eaxx04'
student@ubuntu:~$ objdump -M intel --start-address=0x4005f6 -d overlap
objdump: 'overlap': No such file
student@ubuntu:~$ objdump -M intel --start-address=0x400612 -d overlap
objdump: 'overlap': No such file
student@ubuntu:~$ echo "## Screenshot by Kevin Harianto 991602128 [`date +"%F %T
"`] ##"
## Screenshot by Kevin Harianto 991602128 [2024-07-30 06:02:22] ##
student@ubuntu:~$
```

4.2 QUESTIONS

No Questions