US Army Cyber School 2018

Reverse Engineering - Activity

```
1001 | 10001100 1060 |
                                                              1 (101010 0100011011101111 100100
                   officials: 1100 101 10001011 010
             JUL 200 016 T1C+1 0100
                                   10100101 9
                                                    ١ 101 0 00: 11010 1000 100 1001 1001 11010100 10001101 01100 1
 aut 01010010 01160011 0000010° ...J0C1
                                   10011.
.0011 11010010 100001111 00° → 1 F C
                                                  1: 7.0 00 01011111 00111001 01100111 11111100 011!
       10001 00100101 111.
       00010111 01000011 00011100 11100010 10011100 011000! C
                                             J00011 01000111 000111 '
                                             {111001 00111001 110
      101000 00010110 001
                                             00100 01001000 00110111 00100001 01000101 01010001 0010.
       11010 10001100
                                           1111100 01111101 011.1101 1.2:1001 11001010 11101000 1001.
       511040
                                          10011100 01100010 0101,111 11010
                                                                          0001100 10001101 01
          10116
                                          01001001 11100010 01011c !1 11000
                                          00011111 00001000 01011011 0
                                                                                     1001 (
                                           `1010010 01100011 00000100 00'
                30000000
                                            210012 10000111 00011000 17
                   01100111 UU
                                                    100101 11110001
                                                                                     1i 010 ....
                  J 00110111 001
                  .1 011111101 1101100 a
                                                    911110 111011
                                                                                      91 201 17, 11 00001.
                                                     10101 160106
                  2 01011111 11010011 )
                                                     00111 000111
                   01011011 11000101
                    1011011 0100111
                                                     11001 110010 . 03
                                                                                              11 100b, J0
                      20100 0000101
                                                     10110 0011
                                                                                           Jul 101 01100111
                      11000 10011
                                                      1100 1010
                                                                                          1001000 00110111
                                                      100 010
                      10001 0011
                                                                                          1111101 011111101
                       1111 101
                                                                                          11000 91011111
                       (0.0000)
                                                                                                 :0110
                       111C
                       101!
                        010
                                                       ... 01011011 11050101 10001011 01000011 01000.
               . 1010 100.3010 10000010
                                             3311 10000000 01100110 ......31010010 01100011 00000100 00001011 100
                                          31010010 01100011 00000100 00001011 10011100 001010C
```

x64 Stack Practice

Introduction

In this activity, you will be stepping through x64 ASM source code. You will need to show everything that is happening on the stack. As well as, how the registers are being affected. The following source code will be broken down over the next few pages:

```
main:

mov rax, 5
push rax
mov rax, 1
push rax
pop r12
pop r13

loop:
add r12, 1
cmp r12, r13
jl loop

mov rax, 0
ret
```

The first portion of this code on page 3 has been completed for you as an example.

Activity

main: mov rax, 5		
Registers	Stack	RSP
rax =5		
Notes F is mayord into ray		
5 is moved into rax. The rsp remains at the		
highest postion in memory on the stack.		
		rsp

Activity		
push rax		
Registers	Stack	RSP
Notes		

Activity		
mov rax, 1		
Registers	Stack	RSP
Notes		

Activity		
push rax		
p som rom		
Registers	Stack	RSP
Notes		

Activity		
pop r12		
Registers	Stack	RSP
registers		
Notes		

Activity		
pop r13		
Registers	Stack	RSP
Notes		
Notes		

Activity

```
loop:
add r12, 1
cmp r12, r13
jl loop
```

Provide an explanation of what is happening in this code segment to include how the registers are affected and what actions are taken.

Activity		
mov rax, 0		
Registers	Stack	RSP
Notes		

Activity		
ret		
Registers	Stack	RSP
Notes		
Notes		