

1a) Prove XORing an operand with itself changes the operand to zero.

XOR Truth Table

a	b	result
0	0	0
0	1	1
1	0	1
1	1	0

Therefore: if a bit is zero and you execute XOR bitwise with another bit that is zero, the result is zero. Likewise, if the two bits are one, the result is zero. The result is only one if the two bits are different. That is to say if the first bit is one, the second must be zero for the XOR to result in one, and vice versa.

ex: bitwise XOR of Hex: 0x6 \Rightarrow binary: 0110

Start bit	0	1	1	0
Second bit	0	1	1	0
Result	0	0	0	0

1b) we can show this in GDB by observing value saved in the eax register after using `xor eax, eax` (assuming a non-zero value has already been placed in eax).

2a) After observing a simple:

```
mov eax, 11  
test eax, 1
```

in GDB, I noticed

the only real change was to the flags section in GDB's register watching panel. After some research I found you can use code to trigger off flags; I would use those triggers with a jump to indicate even or odds to the user. Could simulate a 50/50 probability/coin flip.

Flow charts

The XOR op is sequential with no decision making, resulting in a linear flowchart. The "test" op has one Yes/no decision.

1) XOR zeroing

Declare 2 variables to store the before and after XOR operation values of eax in section .bss:

"init_eax" and "after_xor"

Set eax to an arbitrary, non-zero value $\Rightarrow \text{mov eax, 15}$

Store eax value in "init_eax"

Move "init_eax" into ebx and check to see what the value is: should be 15

Store new eax value into "after_eax"

Perform XOR zeroing operation
 xor eax, eax

Move "after_xor" into ebx and check to see what the value is: should be 0

Terminate program
 mov eax, 1
 int 0x80

2) TEST

Move an even or odd number into eax

Run a "test" operation on eax with one:
 test eax, 1

In GDB, observe the flags

Terminate program

eax value is Even

Is Zero flag set?

eax value is Odd

No

Yes

Challenges:

The xor zeroing was straight forward and simple as I was exposed to the concept during the w6 activity as a possible solution to a problem I was having with mul/imul. But I was confused as to what "test" was actually doing. I noticed the flags section was changing and looked it up. After some reading on stack overflow I found out that we can interrogate the flag status, in this case the zero flag (ZF), to use "test" in a meaningful way. For the case of determining even/odd, I'd check if the zero flag got set after "test eax, 1". If the ZF is set it means that the LSB of eax was not one, therefore it is True ($ZF=1$) that the eax LSB is 0. So if ZF is 1, eax LSB is 0. If ZF is 0, eax LSB is 1.