CS 208

Wed. Oct 11, 2023

Test #3

-55_{ten}

1. Write 55 in binary
2. Write that as a 32-bit #
3. Complement + add 1
4. Write result in hexdecimal

4D UTF-16 LE BE

U+2295

22 95

byte1 byte2

95 22

byte1 byte2

#5

① Think about the # of bytes in various C types (int — 4 bytes, char* — 8 bytes)

② ints are stored as two's complement, 32 bits / 4 bytes

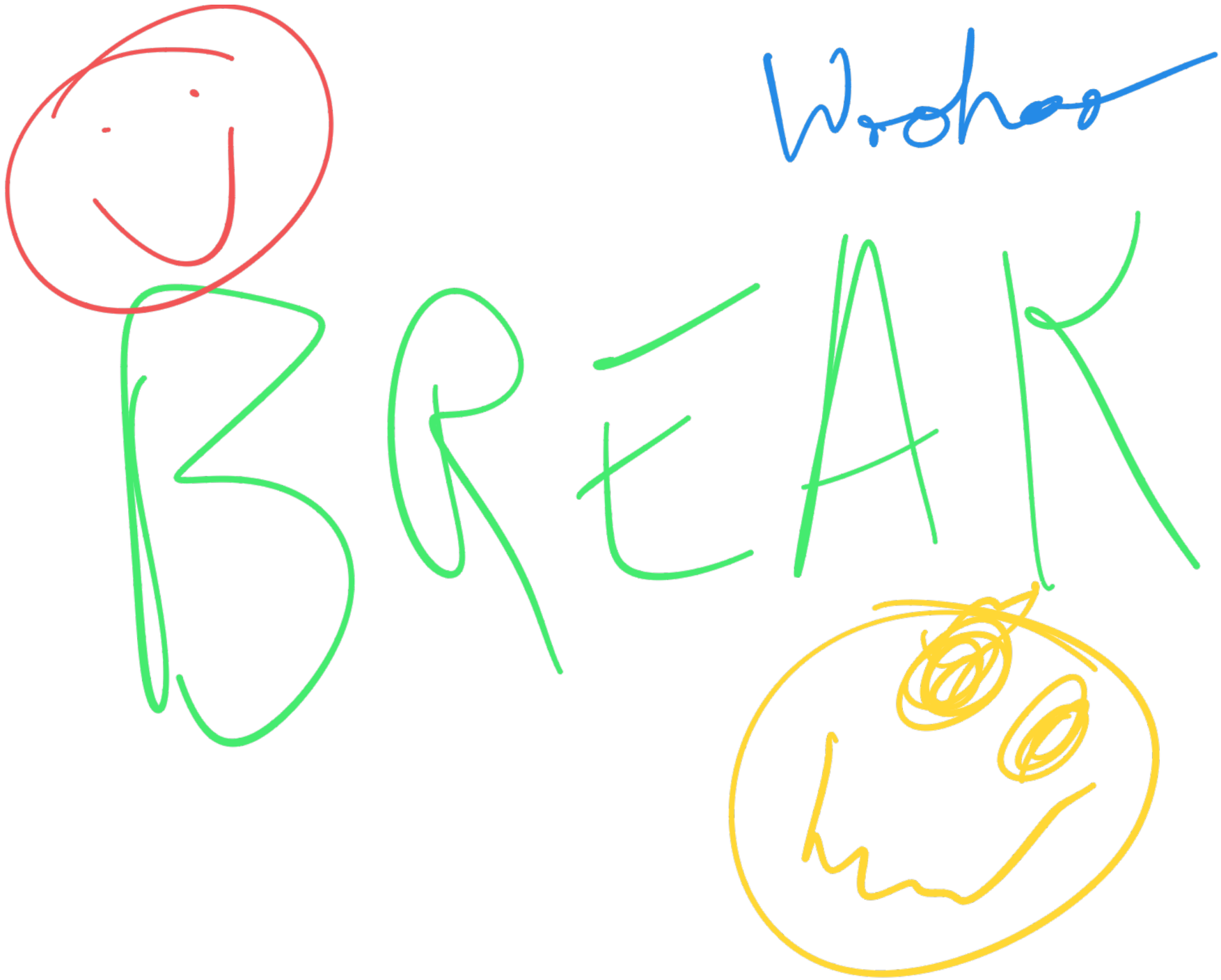
6. %x in printf says "hexadecimal"

ints are 4 bytes

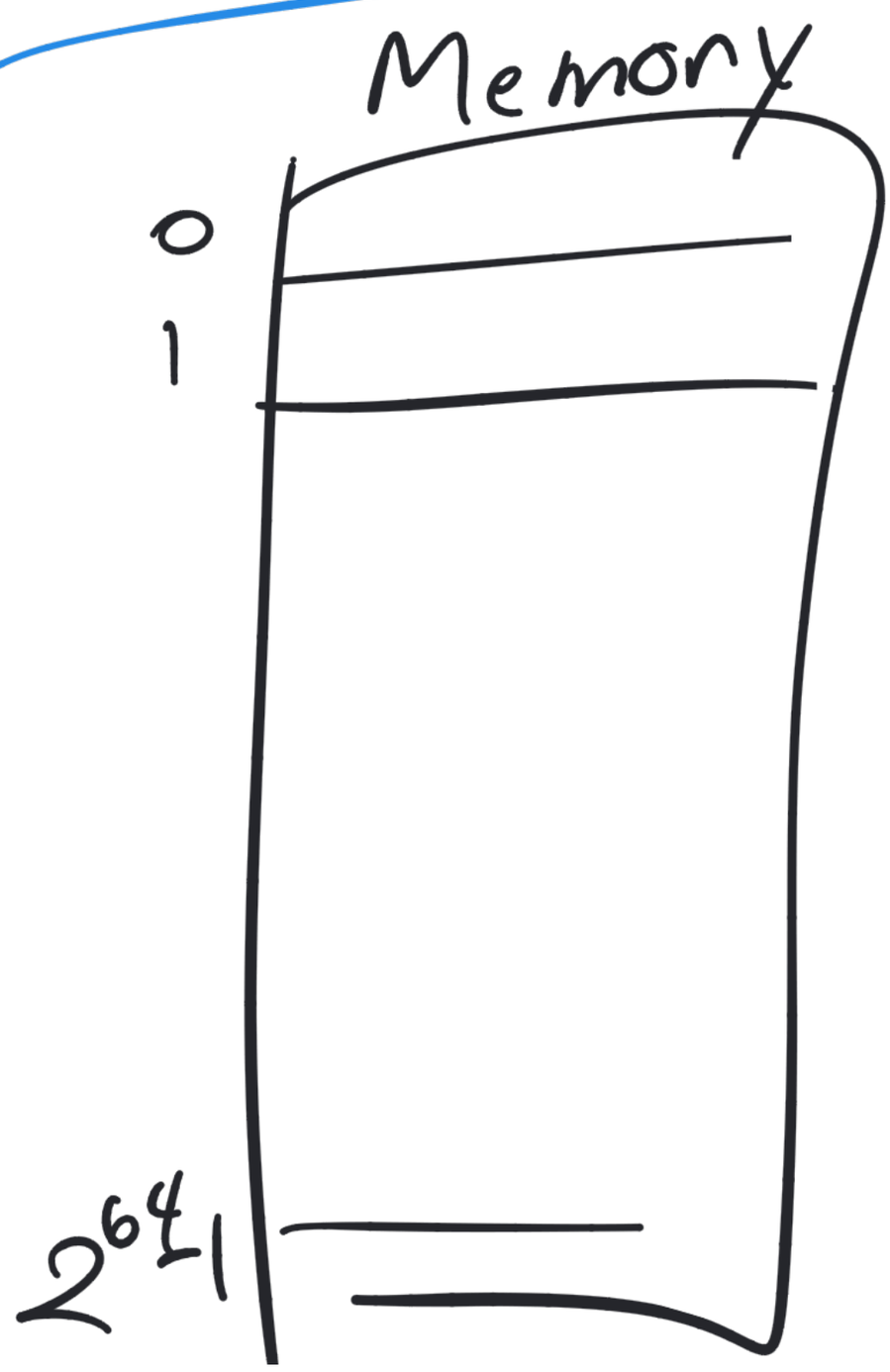
a ~ 1011 ~ 0000...0000...01011

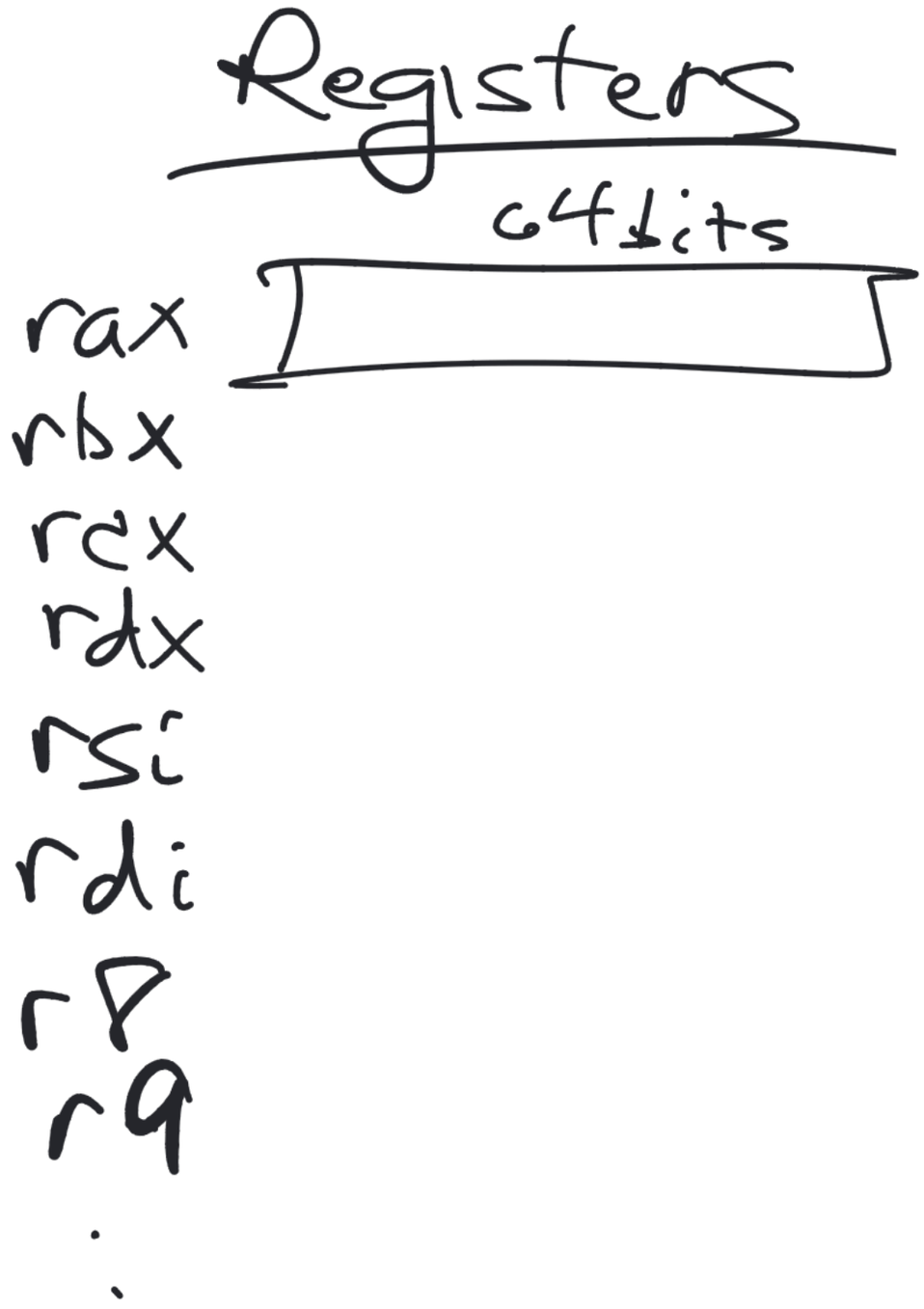
≥2 bits

~a ~


Woohoo
BREAK

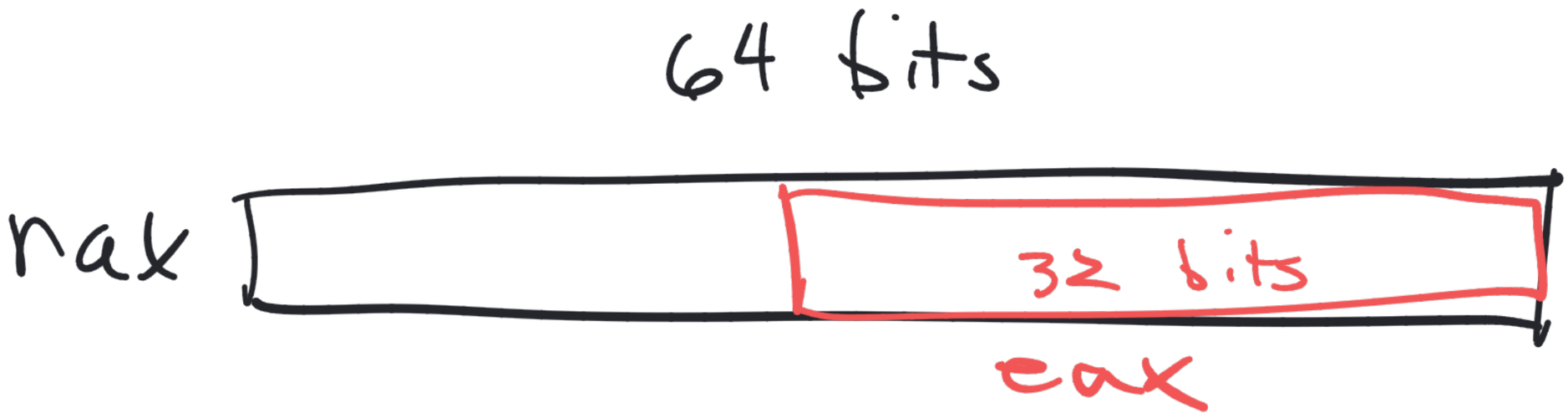
x86-64





Function-calling

1. Caller puts parameters in rdi, rsi, rdx, ...
2. call name_of_function
3. function assumes params are in rdi, rsi, ...
4. ...
5. fn puts return value in rax
6. ret


64 bits
rax
32 bits
eax

EFLAGS

test %edi, %edi

① Compute %edi & %edi

② IF result < 0
(EFLAGS) SF = 1
else (EFLAGS) SF = 0

} store the sign of the answer in SF

③ If result == 0
ZF = 1
else
ZF = 0

} store zeroness of answer in ZF

if (a < 0) {

}

test %edi, %edi

sets value of SF

js .L3

js : jump if sign bit (SF) is 1

cmpl %edi, %esi

%edi — %esi

set SF, ZF