CS 208
Fri., 20 Jan 2023
int j = -1;

Two's complement of 0:

printf("0x%"x", j);

0xffffffff
unsigned int k = -1;

Better style/readability:
unsigned int k = 0xffffffff;
char ch = 0x41;
int k = ch;

printf("0x%08x\n", k);

0x00000041
char ch = 0xCE;
int k = ch;

printf("0x%08x\n", k);
0xfffffffff

sign extension during promotion
1. "Type promotion"

\[ k = \text{ch}; \] (copy the 

2. "Sign extension"

fill in the extra bytes)
char ch = 0x41
printf("%d\n", ch);

printf wants an int

0 Promote ch to int (with sign extension)
1 Print the int

Result: 65
char ch = 0xCE;
printf("%d\n", ch);

printf wants an int

1. Promote ch to int (with sign extension)
2. Print the int

Result: -50
$0 \times CE = -50$

$1100 \ 1110$

neg. which neg?
let's negate

$0011 \ 0001$

$+ 1$

$0011 \ 0010 = 32 + 16 + 2 = 50$

8-bit two's comp.
One more thing after class

>> also does

Sign extension

When you shift right, the bits that flow in from the left depend on the original leftmost bit.
\( \text{int} \ k = 20 \);  
\( \text{int} \ j = (k \gg 2) \);  

(“hey! shift right by 2 bits divides by 4”)
```c
int k = -20;
int j = (k >> 2);
```

Sign extension again! (Division by 4 again.)
What about unsigned?

promotion still happens.

unsigned char ch = 0xCED;
unsigned int k = ch;

Sign extension does not.

All extensions in

= or >> are 0 bits.