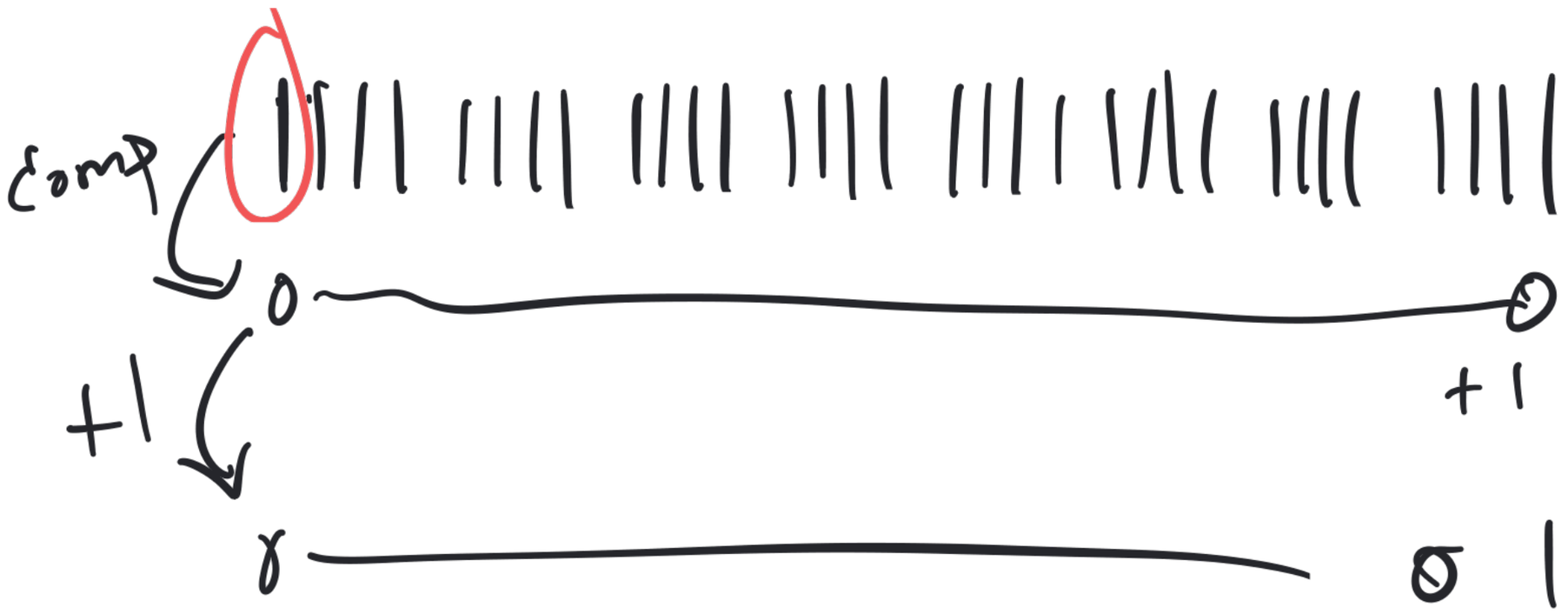
CS 208

Fri., 20 Jan 2023

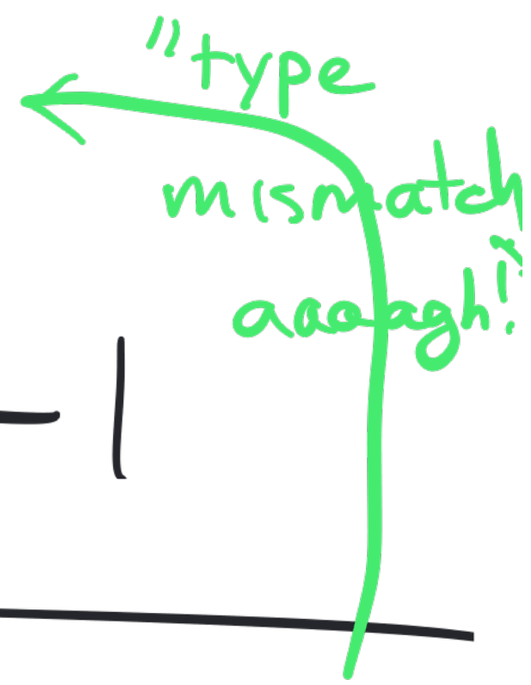
```
int j = ~1;
```



Two's comp Negate to discover that all 1's is -1

```
printf("0x%x", j)
0xffffffff
```

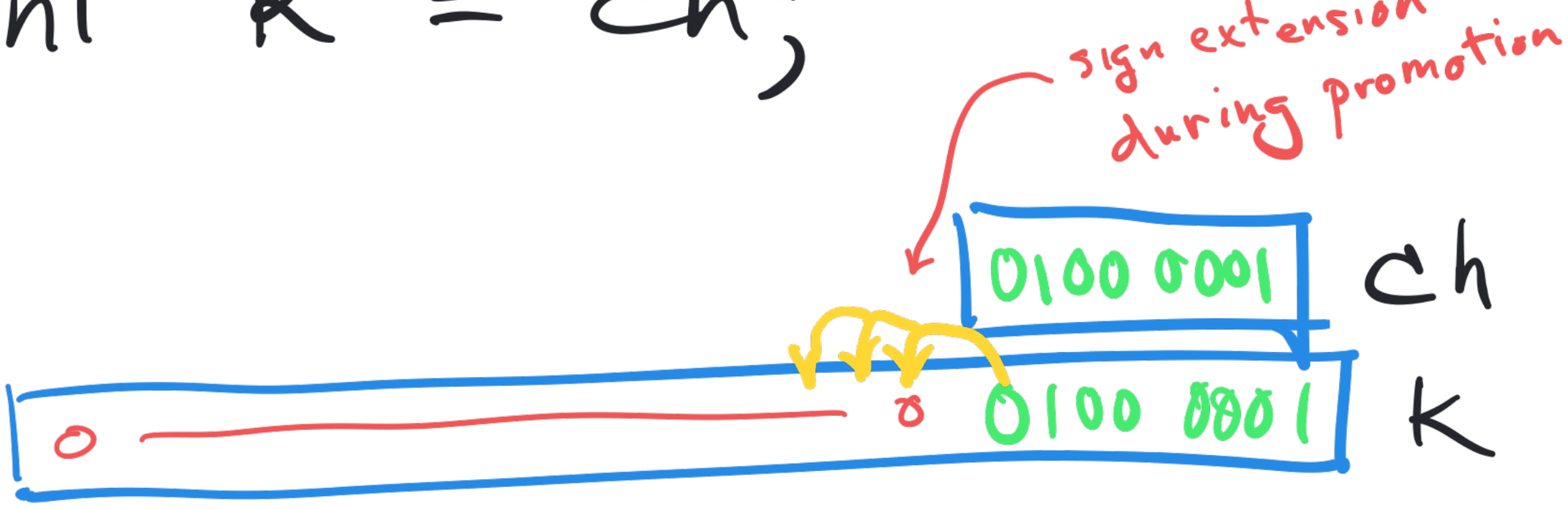
```
unsigned int k = -1;
```



Better style/readability

```
unsigned int k = 0xffffffff;
```

```
char ch = 0x41;
int  k  = ch;
```




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

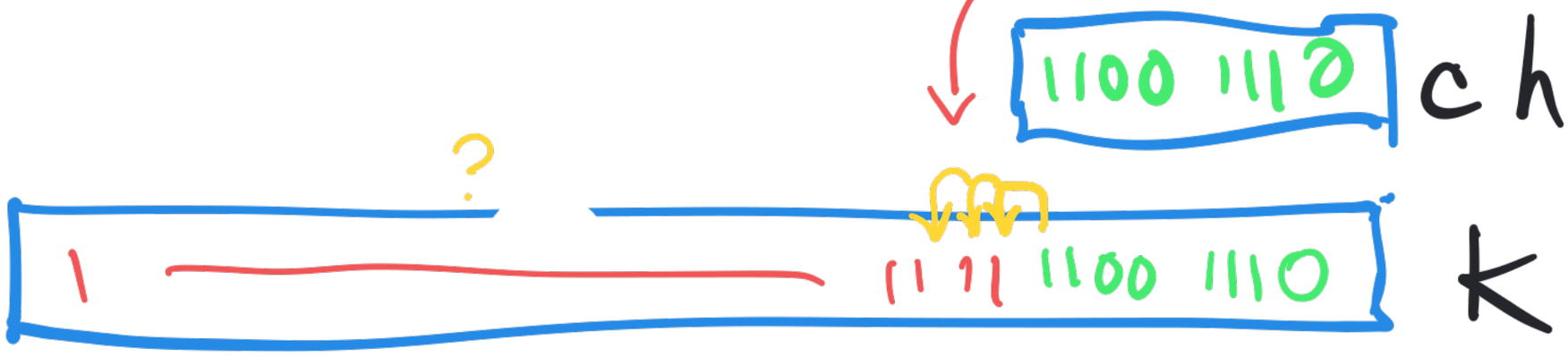
0x00000041

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

(-50)

(-50)

sign extension during promotion




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

0xffffffce

1. "Type promotion"

k = ch; (copy the ch + fill in the extra bytes)

2. "Sign extension"

```
char ch = 0x41
printf("%d\n", ch);
```

printf wants an int

① Promote ch to int (with sign extension)
② 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

0xCE = -50

1100 1110

neg.

which neg?

let's negate

8-bit two's comp.

0011 0001

\+ 1

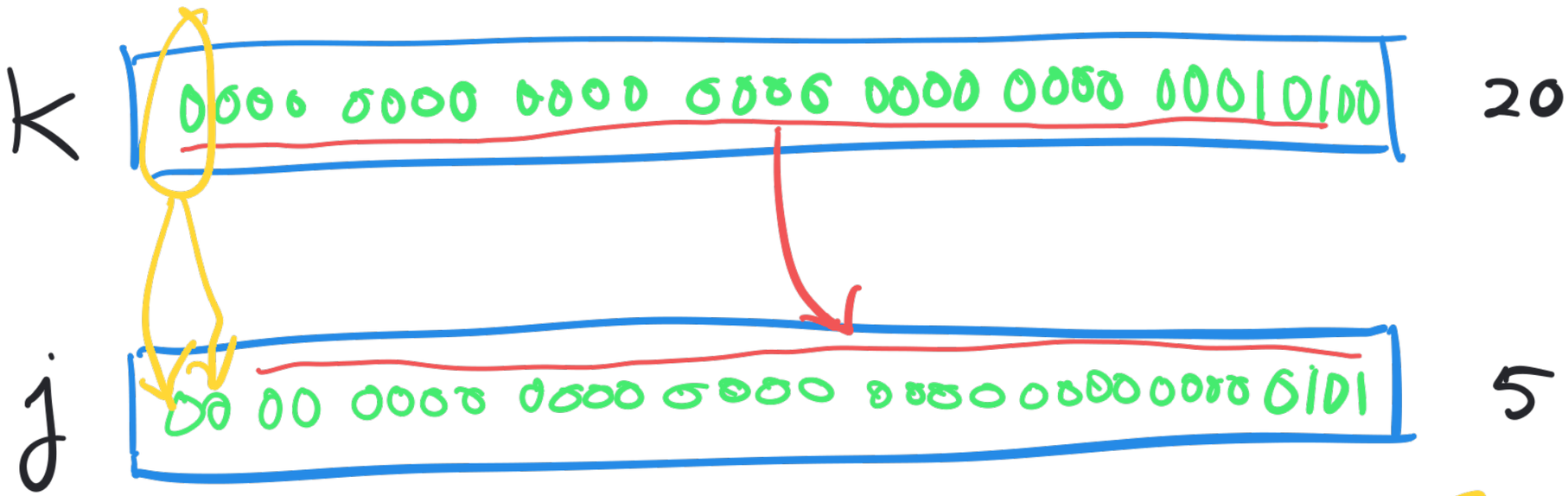
0011 0010 = 32+16+2 = 50

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

```
int k = 20;
int j = (k >> 2);
```



("hey! shift right by 2 bits divides by 4")

```
int k = -20;
int j = (k >> 2);
```

What about unsigned?

promotion still happens.

```
unsigned char ch = 0xCE;
unsigned int k = ch;
```

sign extension does not.

All extensions in

= or >> are 0 bits.