clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Bit Fiddling

Updates

int *temp;

fread(temp.. ..)

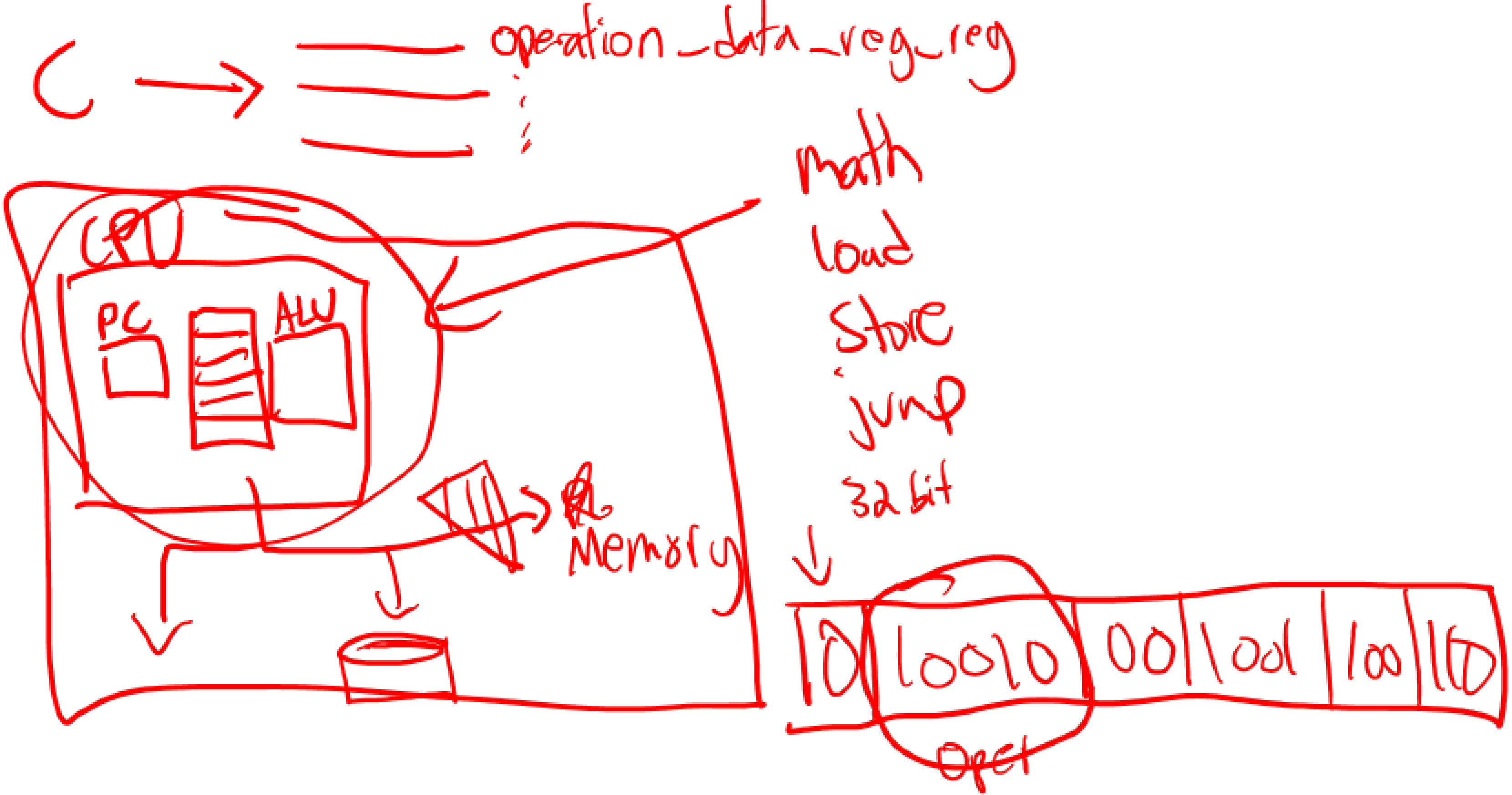
1. Exam 1 scores are released.
    a. Come to office hours for clarity! A few misconceptions that will most likely come up again.
    b. Submit regrades ASAP via Prairie Learn
2. MP 3 - PNG due today
3. MP 4 - UTF-8 out today (due next tuesday)

Agenda

CPU Review

1. Bit fiddling
    a. Bit shifting
    b. Bit logic operations
2. Bit Mask
3. MP4 - UTF-8
4. Sets

Assembly and the CPU



Bytes Review

info stored in 1's 0's bits

8 bits = 1 byte

↳ in Hex 1 byte ↙

0b0010 1001 = 0x29

chars – 1 byte

int – 4 byte

97 → 'a' → $\boxed{\text{0b0110 0001}}$ → 0x61

How many bits are in a char?

1 byte = 8 bits

clicker.cs.illinois.edu

Q2

~Code~
340

How many bits are in an int?

4 bytes

8 bits

= 32 bits

clicker.cs.illinois.edu

Q3

~Code~
340

Bit Fiddling in C

1. Bit shifting << or >>

2. Bit operations (AND OR XOR NOT)

Bit Shifting

byte → x = 0b0100 1100

0's

x = x << 1

*use unsigned

x = 0b1001 1000

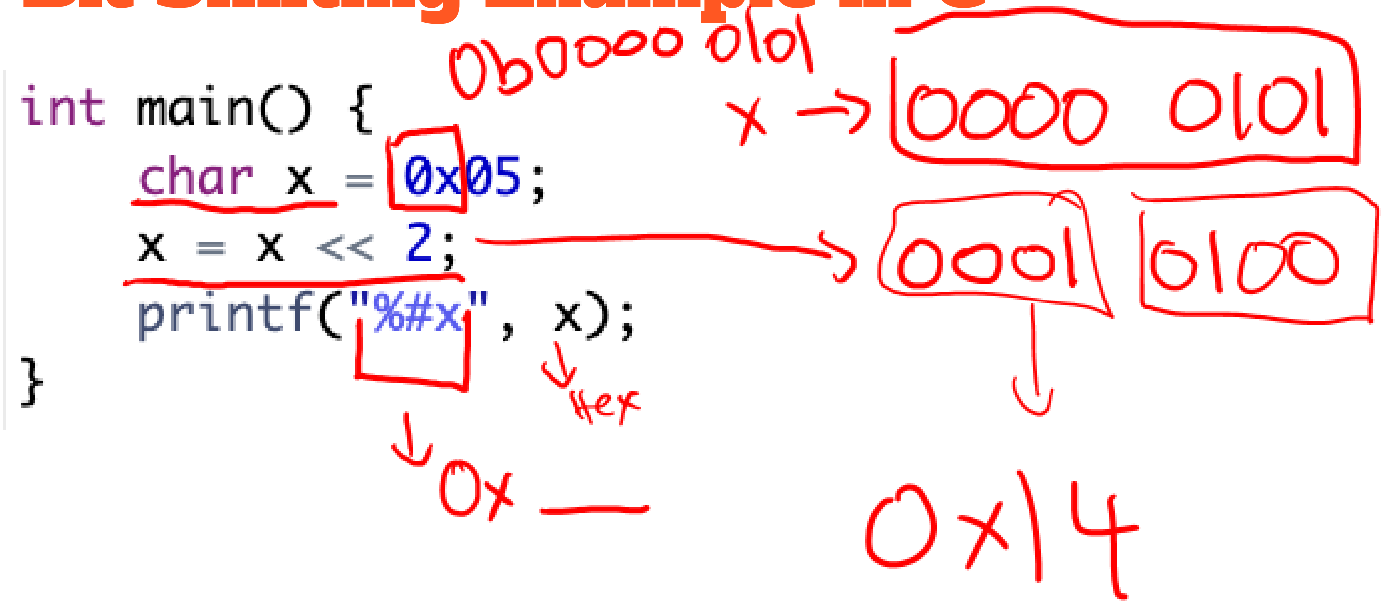
→ 0x56

>>2

x = x >> 1

0b0010 0110

Bit Shifting Example in C



```
int main() {
    char x = 0x05;
    x = x << 2;
    printf("%#x", x);
}
```

What prints?

```
int main() {
  unsigned
    char x = 0x1E;
    x = x >> 3; →
    printf("%#x", x);
}
```

0001 1$\boxed{110}$

000 00 11

010

10

0x3

0x03

0x3

0x03

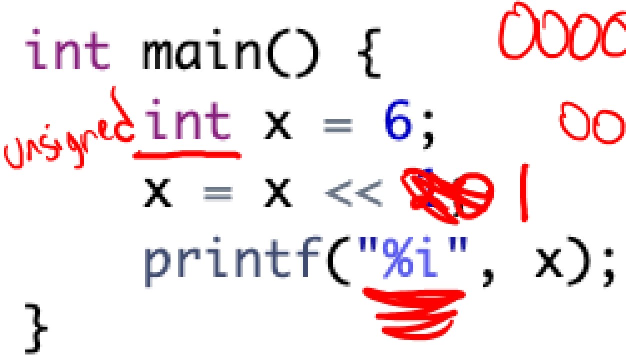
clicker.cs.illinois.edu

Q4

~Code~

340

What prints (challenge)?



```
int main() {
    int x = 6;
    x = x << 1
    printf("%i", x);
}
```

unsigned

0000 0000 0000 0000

0000 0000 0000 0110

0 ← 1100

= 12 in decimal

clicker.cs.illinois.edu

Q5

~Code~
340

Bit Operations

AND	&
OR	|
XOR	^
NOT	~ ← fonts
	-

1001 & 0001

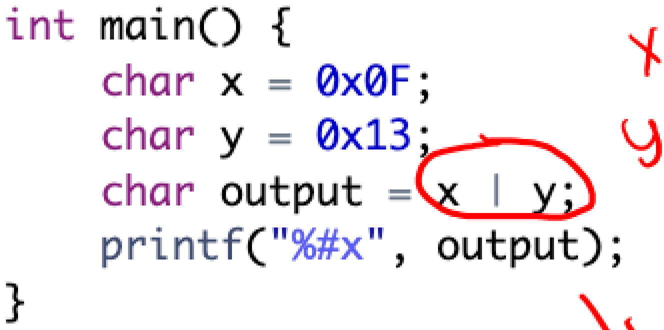
```
1001
0001
----
0001
```

1001 ^ 0001

```
0001
1001
----
1000
```

~1001 = 0110

Bit Operations Example in C



```c
int main() {
    char x = 0x0F;
    char y = 0x13;
    char output = x | y;
    printf("%#x", output);
}
```

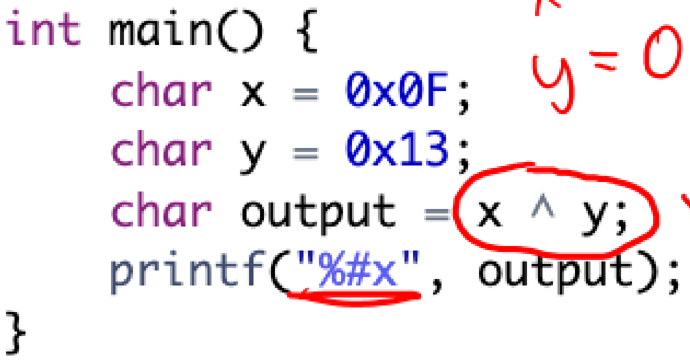
x=0000 1111

y=0001 0011

0001 1111

0x1F

What prints?



```
int main() {
    char x = 0x0F;
    char y = 0x13;
    char output = x ^ y;
    printf("%#x", output);
}
```

x= 0000 1111
y = 0001 0011

XOR

output=0001 1100

0x1C

clicker.cs.illinois.edu

06

~Code~
340

What 10 bit value does this produce?

```
((~0) << 3)
```

~[0000 0000 00]

← 3

1111 1111 11

11 1111 1000

```
char x = 5
x = ~x;
```

0000 0110

1111 1001

```
int x = 5
x = ~x;
```

0000 0000 0110

1111 1111 1111 1001

clicker.cs.illinois.edu

Q7

~Code~

340

What 10 bit value does this produce?

clicker.cs.illinois.edu

Q8

~Code~
340

$$(((\sim 0) << 3) \overset{XOR}{\wedge} ((\sim 0) << 6)) = y$$

11 1 11 1 000   11 1 00 0 000

$y = 00\ 00\ 11\ 1\ 000$

I want just the middle 4 bits from a byte to remain.

$y \& x$

$y = 0\ 0\ 1\ 1\ \ 1\ 0\ 0\ 0$

$x = X_0\ X_1\ X_2\ X_3\ \ X_4\ X_5\ X_6\ X_7$

$0\ 0\ X_2\ X_3\ \ X_4\ 0\ 0\ 0$

How would I get only the 8 smallest order bits from from an int x?

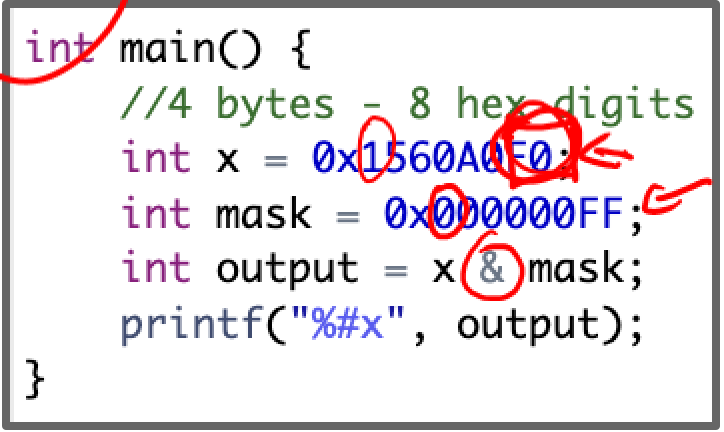
right

clicker.cs.illinois.edu

09

~Code~
340

A

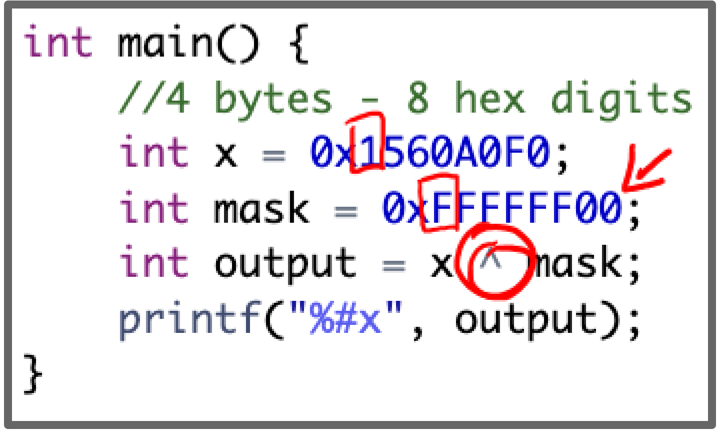


```
int main() {
    //4 bytes - 8 hex digits
    int x = 0x1560A0F0;
    int mask = 0x000000FF;
    int output = x & mask;
    printf("%#x", output);
}
```

want 0xF0

0001 ^
1111
1110

B



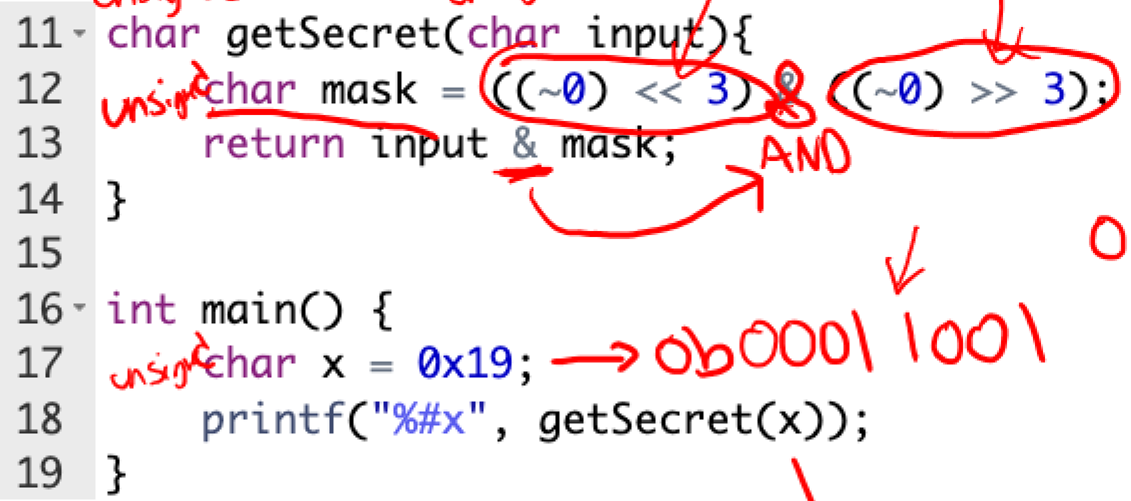
```
int main() {
    //4 bytes - 8 hex digits
    int x = 0x1560A0F0;
    int mask = 0xFFFFFF00;
    int output = x ^ mask;
    printf("%#x", output);
}
```

What would print (challenge)?

clicker.cs.illinois.edu

Q10

~Code~
340



```
char getSecret(char input){
    char mask = ((~0) << 3) & ((~0) >> 3);
    return input & mask;
}

int main() {
    char x = 0x19;
    printf("%#x", getSecret(x));
}
```

unsigned

unsigned

unsigned

unsigned

1111 1000

0001 1111

AND

0b0001 1001

0001 1000

char mask = 0x18;

0001 1000

0x18

But why? Two Examples!

1. UTF-8

2. Bit Sets

MP4 - UTF-8

Char - 1 byte - 8 1's 0's

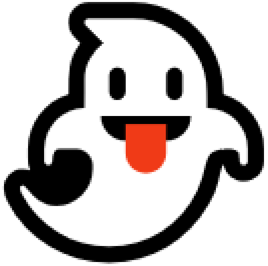
127
0- a, b '

ASCII - number -> character

Unicode - bigger ascii table defines Code Points 5,603

int, -> number ->

UTF-8 - 1-4 bytes
1-4 chars



UTF-8

Code Point - number → unsigned int → represents a symbol ☺ 'a'

UTF-8 - 1-4 bytes representing a code point

(Encoding) Code point -> UTF-8

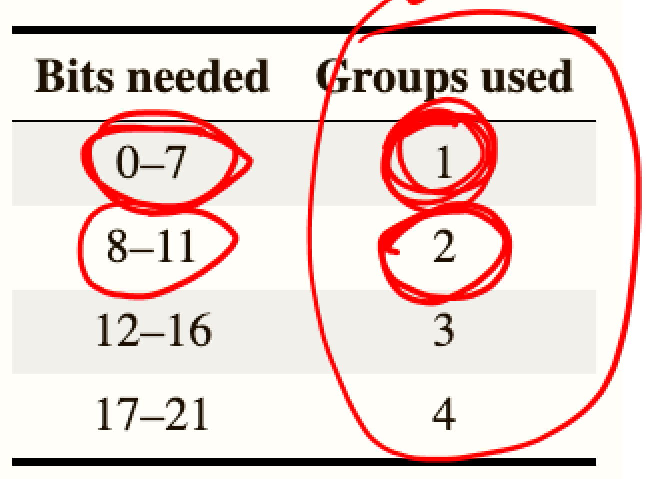
1. count how many bits do we need

Code point = 2 = 2 bits

Code point = 512 = 10 bits

2. figure out how many groups

Bytes/chars



Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

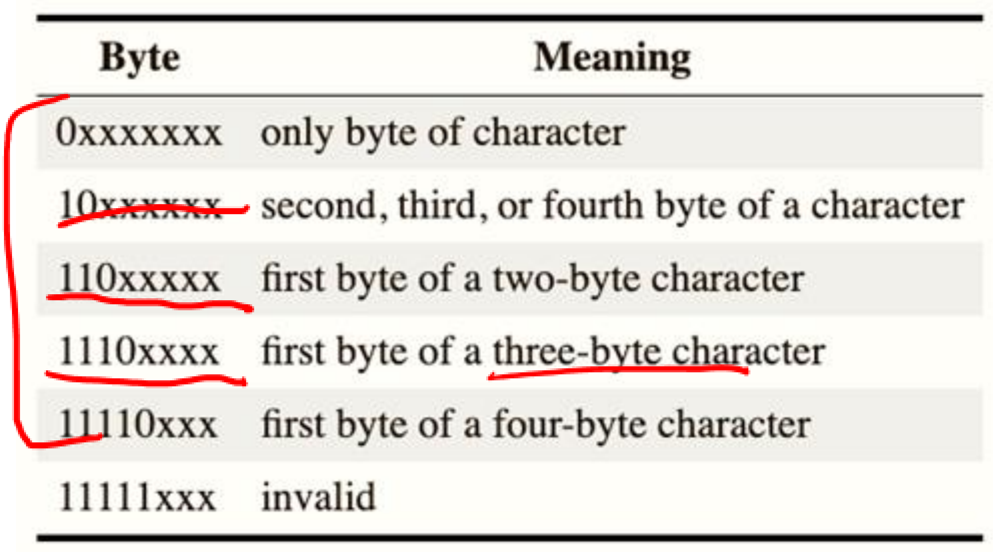
(Encoding) Code point -> UTF-8

1 group = 0xxx xxxx

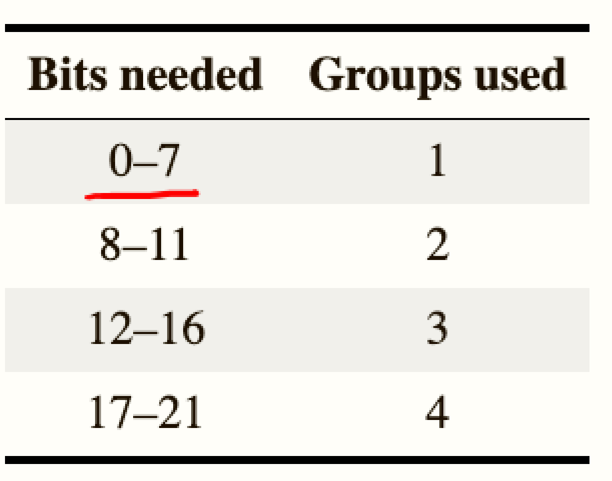
2 grops = 110x xxxx 10xx xxxx

16 x's

3 grops = 1110 xxxx 10xx xxxx 10xx xxxx



Byte	Meaning
0xxxxxxx	only byte of character
10xxxxxx	second, third, or fourth byte of a character
110xxxxx	first byte of a two-byte character
1110xxxx	first byte of a three-byte character
11110xxx	first byte of a four-byte character
11111xxx	invalid



Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

(Encoding) Code point -> UTF-8

U+0BB9 ஹ

1011 1011 1001 (12 bits)

dec = 3001

~~0000~~ 1011 1011 1001

0xE0 0xAE

1110 0000 1010 1110 1011 1001

0xB9

byte 1

Byte	Meaning
0xxxxxxx	only byte of character
10xxxxxx	second, third, or fourth byte of a character
110xxxxx	first byte of a two-byte character
1110xxxx	first byte of a three-byte character
11110xxx	first byte of a four-byte character
11111xxx	invalid

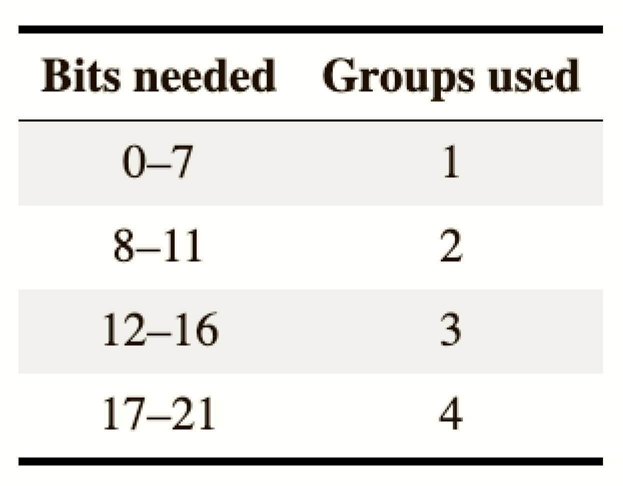
Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

If I need 9 bits to represent a code point, how many bytes will I need to encode it to UTF-8?

clicker.cs.illinois.edu

Q11

~Code~
340



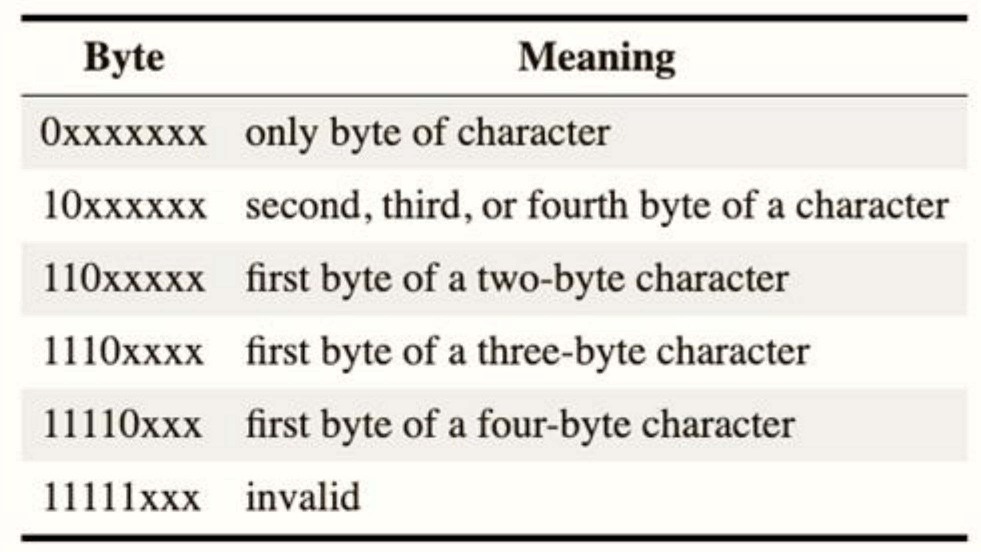
Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

How many X slots are there in a 4-group UTF-8 character?

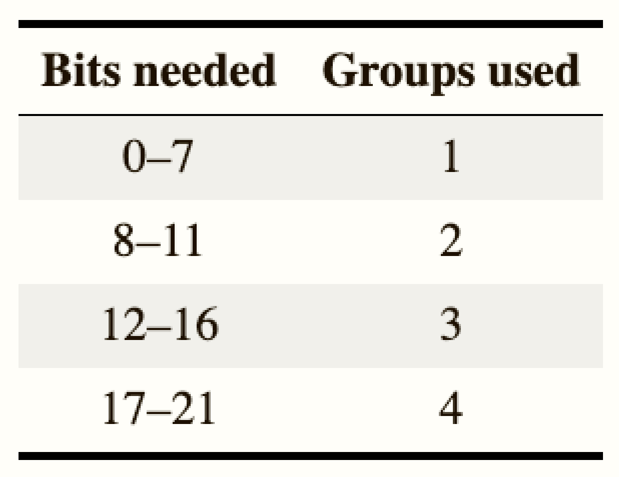
clicker.cs.illinois.edu

Q12

~Code~
340

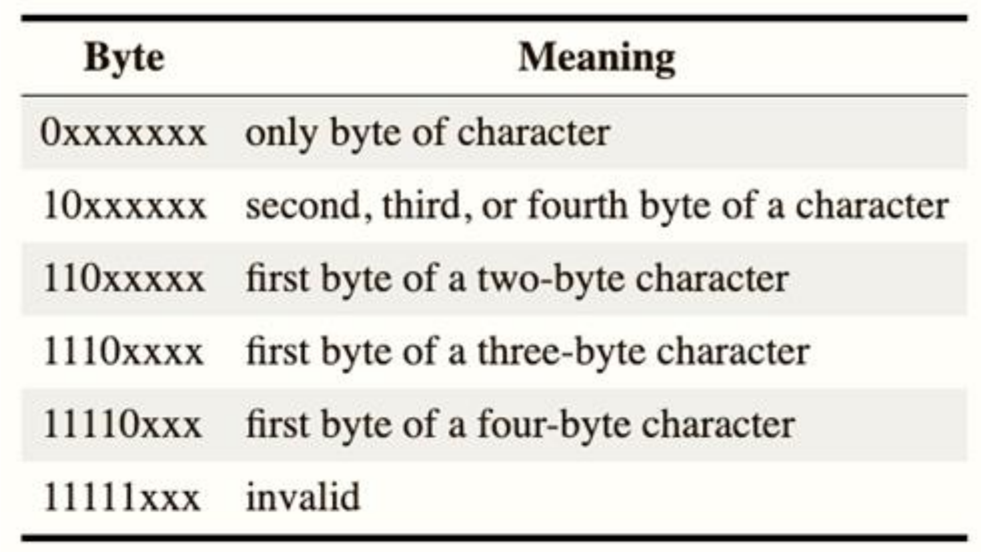


Byte	Meaning
0xxxxxxx	only byte of character
10xxxxxx	second, third, or fourth byte of a character
110xxxxx	first byte of a two-byte character
1110xxxx	first byte of a three-byte character
11110xxx	first byte of a four-byte character
11111xxx	invalid

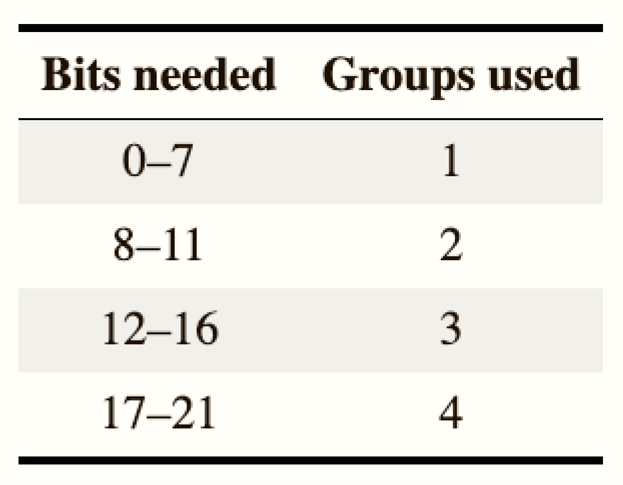


Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

(Decoding) Code point -> UTF-8



Byte	Meaning
0xxxxxxx	only byte of character
10xxxxxx	second, third, or fourth byte of a character
110xxxxx	first byte of a two-byte character
1110xxxx	first byte of a three-byte character
11110xxx	first byte of a four-byte character
11111xxx	invalid



Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

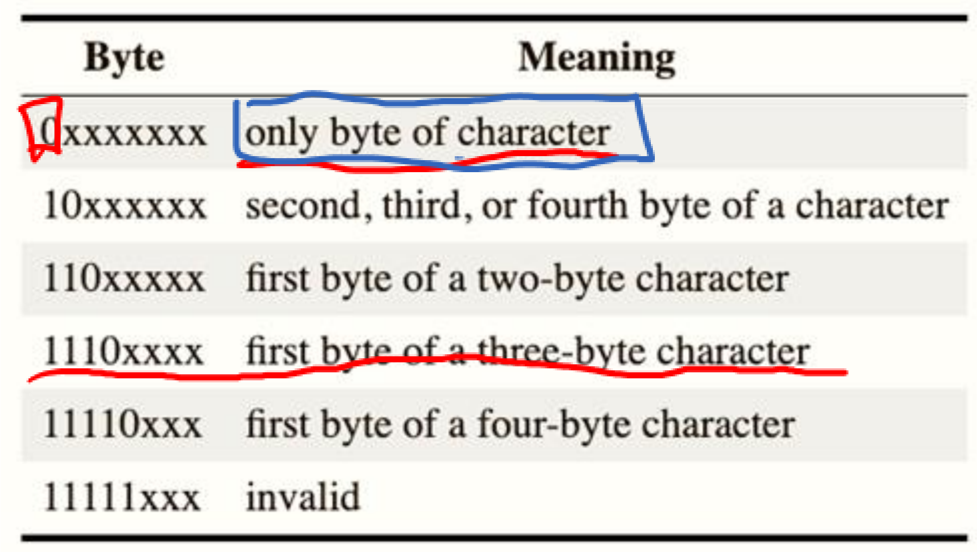
(Decoding) Code point -> UTF-8

'a' 97

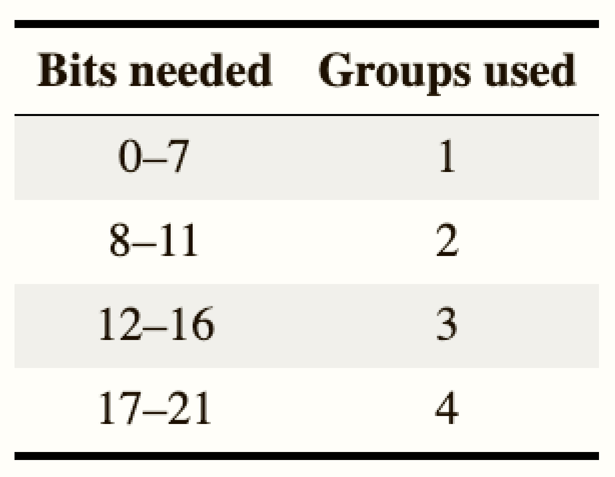
0000 1000 1110 0100 1000 0001 1001 0000

ascii = 8

0100 00 000101 0000 = 16,464



Byte	Meaning
0xxxxxxx	only byte of character
10xxxxxx	second, third, or fourth byte of a character
110xxxxx	first byte of a two-byte character
1110xxxx	first byte of a three-byte character
11110xxx	first byte of a four-byte character
11111xxx	invalid



Bits needed	Groups used
0–7	1
8–11	2
12–16	3
17–21	4

MP4 - UTF-8

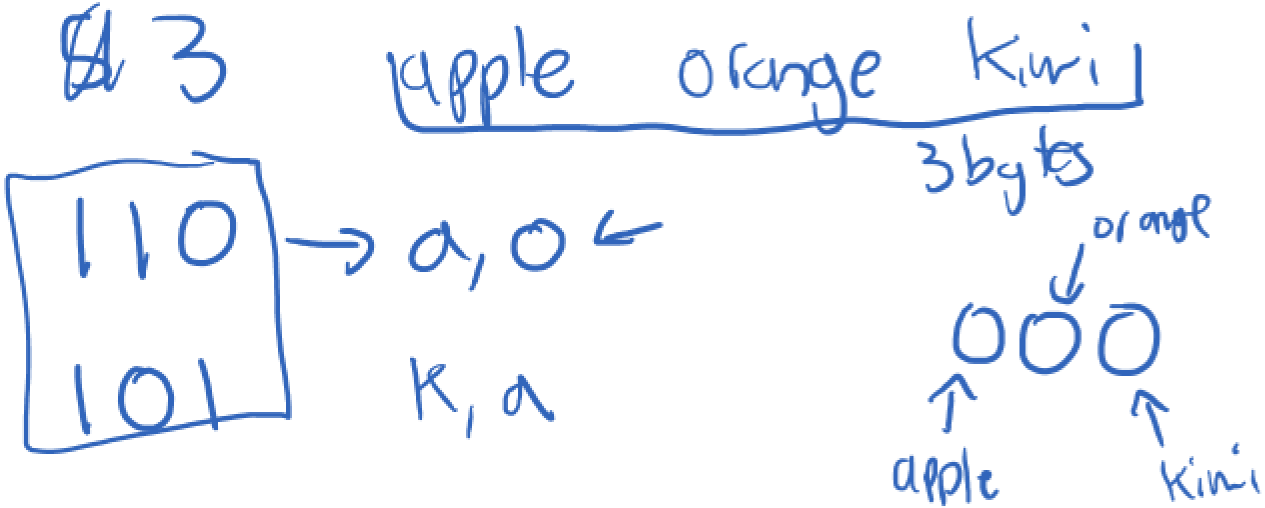
But why? Two Examples!

1. UTF-8

1. Bit Sets

Bit Sets

Set - Collection of things with no repeats or enforced order



How many bytes do I need to represent a set of 16 items?

clicker.cs.illinois.edu

Q13

~Code~
340

What operation would I need to find the union of two fruit basket sets?

clicker.cs.illinois.edu

Q14

~Code~
340

$$\begin{array}{r} 110 \\ \text{or}\ 101 \\ \hline 111 \end{array}$$

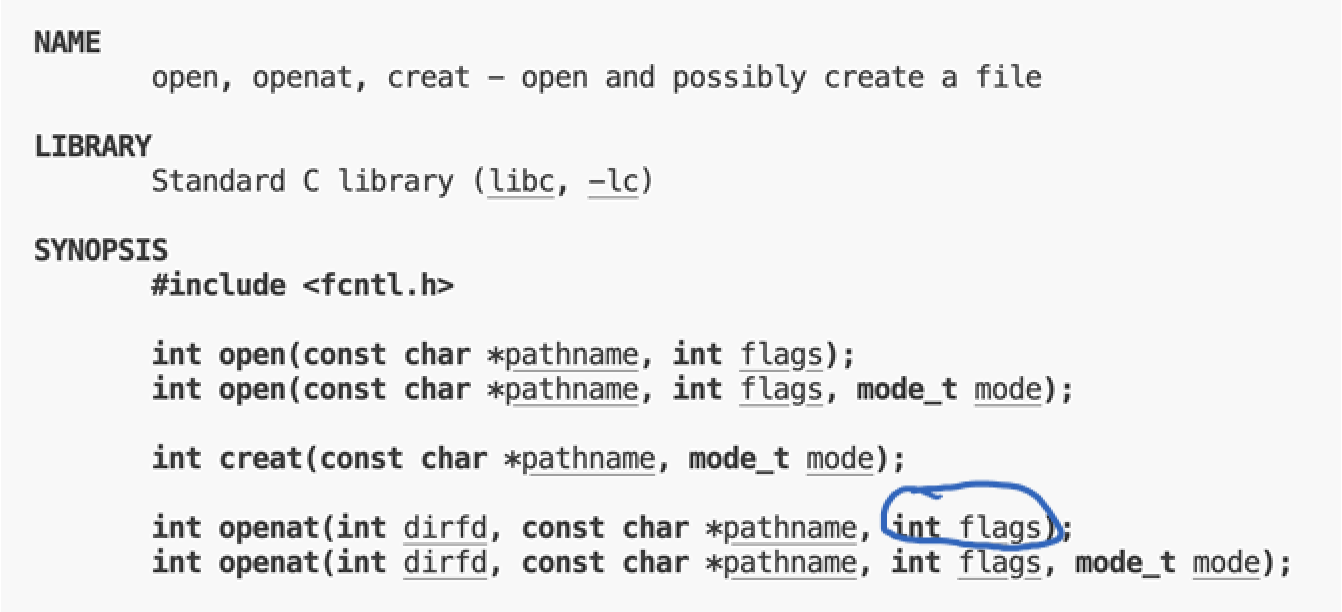
What operation would I need to find the intersection of two fruit basket sets?

clicker.cs.illinois.edu

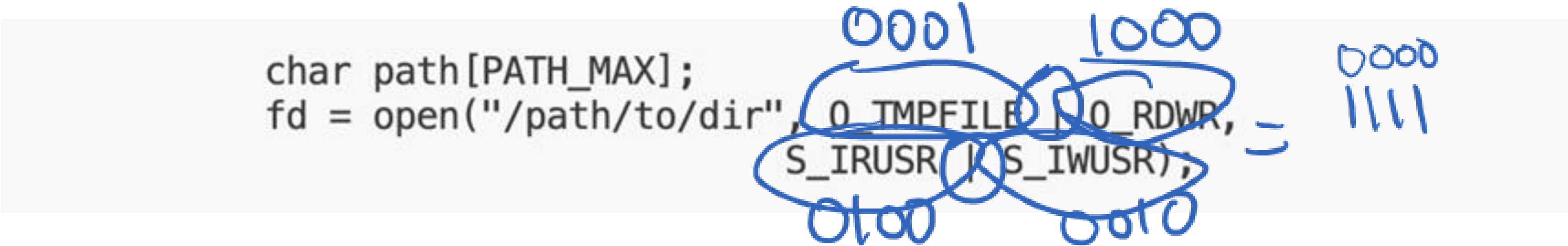
Q15

~Code~
340

Bit Sets - Used as flags



```
NAME
       open, openat, creat - open and possibly create a file

LIBRARY
       Standard C library (libc, -lc)

SYNOPSIS
       #include <fcntl.h>

       int open(const char *pathname, int flags);
       int open(const char *pathname, int flags, mode_t mode);

       int creat(const char *pathname, mode_t mode);

       int openat(int dirfd, const char *pathname, int flags);
       int openat(int dirfd, const char *pathname, int flags, mode_t mode);
```



```
char path[PATH_MAX];
fd = open("/path/to/dir", O_TMPFILE | O_RDWR,
                          S_IRUSR | S_IWUSR);
```

0001 1000

0100 0010

= 0000

1111