Would you rather...

A) be a morning person

B) be a night person

clicker.cs.illinois.edu

Q1

~Code~
340

CS 340

Part I Review and Exam 2 Prep

Updates

1. MP6 Wallet due in 2 weeks

2. Exam 2 next Tuesday (no class Tuesday)

practice problem on PL

Hw 4, 5

MP 4, 5

Agenda

1. Part 1 Review in the context of Exam 2
    a. Assembly and Processors
    b. Virtual Memory
    c. OS and Concurrency
    d. Bit Fiddling
    e. Negatives
    f. Threading

CS 340 - Demystifying Computer Systems

LG: Getting experience working on unstructured, substantial coding projects.

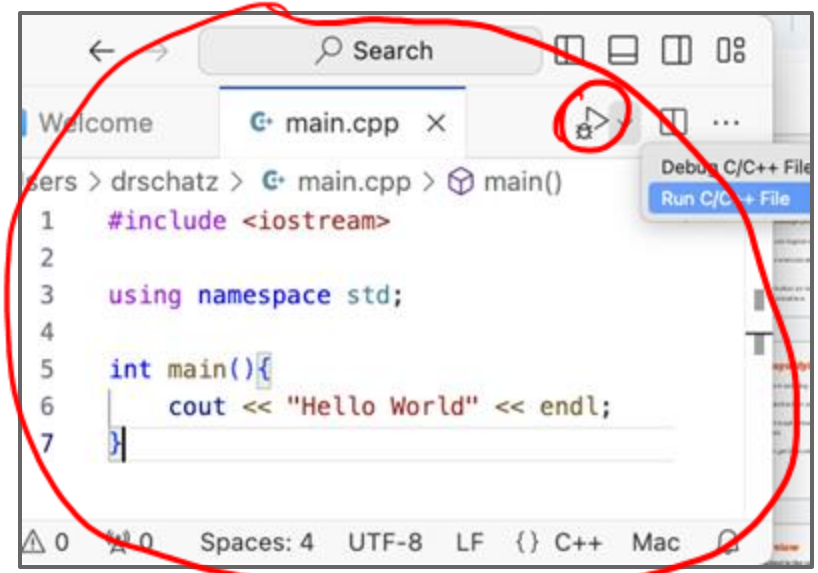
MP

LG: Understanding abstraction and how it relates to computer systems.

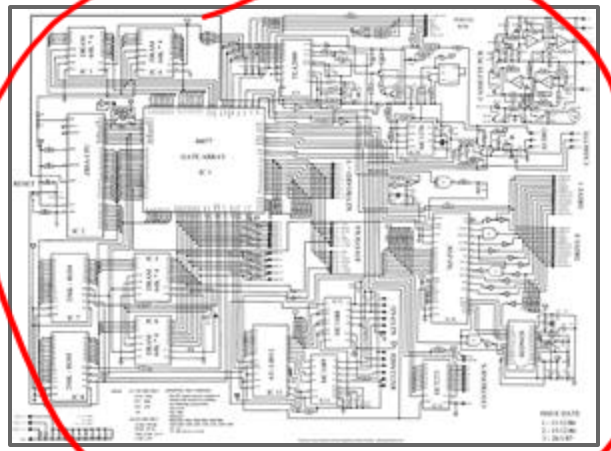
LG: Understanding enough about computer systems to be able to dive deeper on your own if needed.

LG: Learning how we get from electricity to the CS 340 Website displaying your grade.

Computer Layers of Abstraction



Part 1



Which of the following is true?

clicker.cs.illinois.edu

Q2

~Code~
340

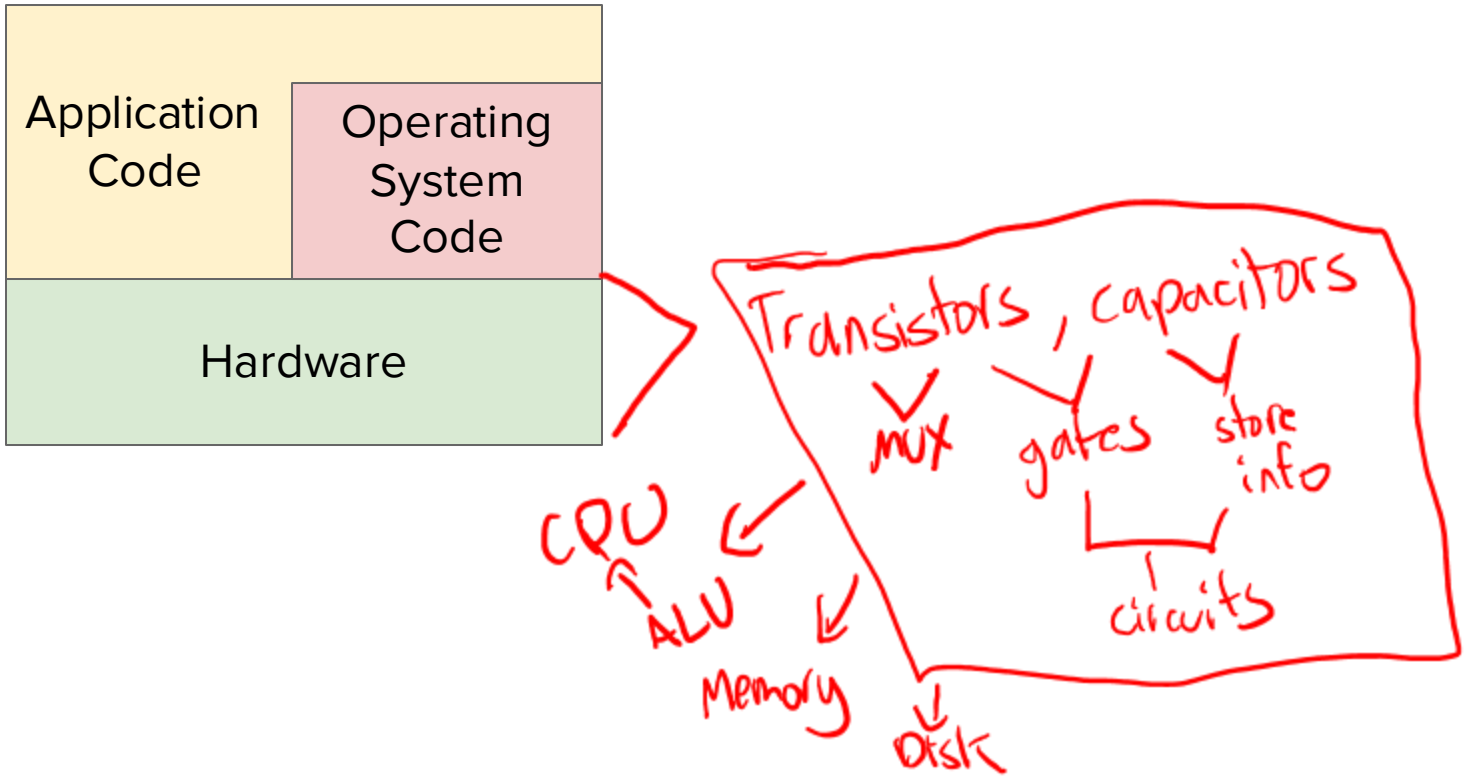
A) To print “hello world” to your screen you can write code in C in VS Code and hit the run button.

B) To print “hello world” to your screen the OS has to manage multiple programs and devices.

C) To print “hello world” to your screen your computer needs to use logic gates and MUX’s.

All of the above

Big Ideas


Application Code
Operating System Code
Hardware
Transistors, capacitors
MUX
gates
store info
circuits
CPU
ALU
Memory
Disk

Explain to a neighbor...

What does it mean to compile a C program? What is the output? Why do we need to compile a program?

clicker.cs.illinois.edu

Q3

~Code~
340

Explain to a neighbor…

Why don’t we just code in straight assembly?

What does a CPU/processor do?

clicker.cs.illinois.edu

Q5

~Code~
340

a) It runs python, C, and other code instructions.

b) It runs 1 assembly instruction at a time using registers and the ALU.

c) It runs many instructions in parallel by utilizing the ALU.

B

Does every user program use virtual memory?

clicker.cs.illinois.edu

Q6

~Code~
340

a) Yes, to provide the illusion of having a lot of space without actually having to provide that much space.
b) Yes, to make it impossible for a process to access another processes information.
c) No, some user programs can run on physical memory as long as we monitor the output.
d) No, some user programs don't need a lot of space to start with so they run on a small part of physical memory.

Explain to a neighbor…

active same time

What is the difference between concurrency and parallelism?

running same time

What are the benefits of each?

faster

clicker.cs.illinois.edu

Q7

~Code~
340

Explain to a neighbor...

x = 5; reg1 - 5 reg3 - 7

→ x = x + 2 reg2 - 2

clicker.cs.illinois.edu

Q8

~Code~
340

What is the difference between a thread and a process? How are they handled differently and similarly by the OS?

A-1 PC^{A1} reg^{A1} $VM \leftarrow A$

A-2 → PC^{A2} reg^{A2}

B-3 PC^{B3} reg^{B2} VM^{B}

Thread

PC, reg

assembly

Why Bits?

hardware easy to tell 1 ↑ 0 ↓ voltage

→ store things with 0, 1 → 8 bits

bit

Byte

↓

0xF7

1111 0111

Bits are Bits – 0, 1

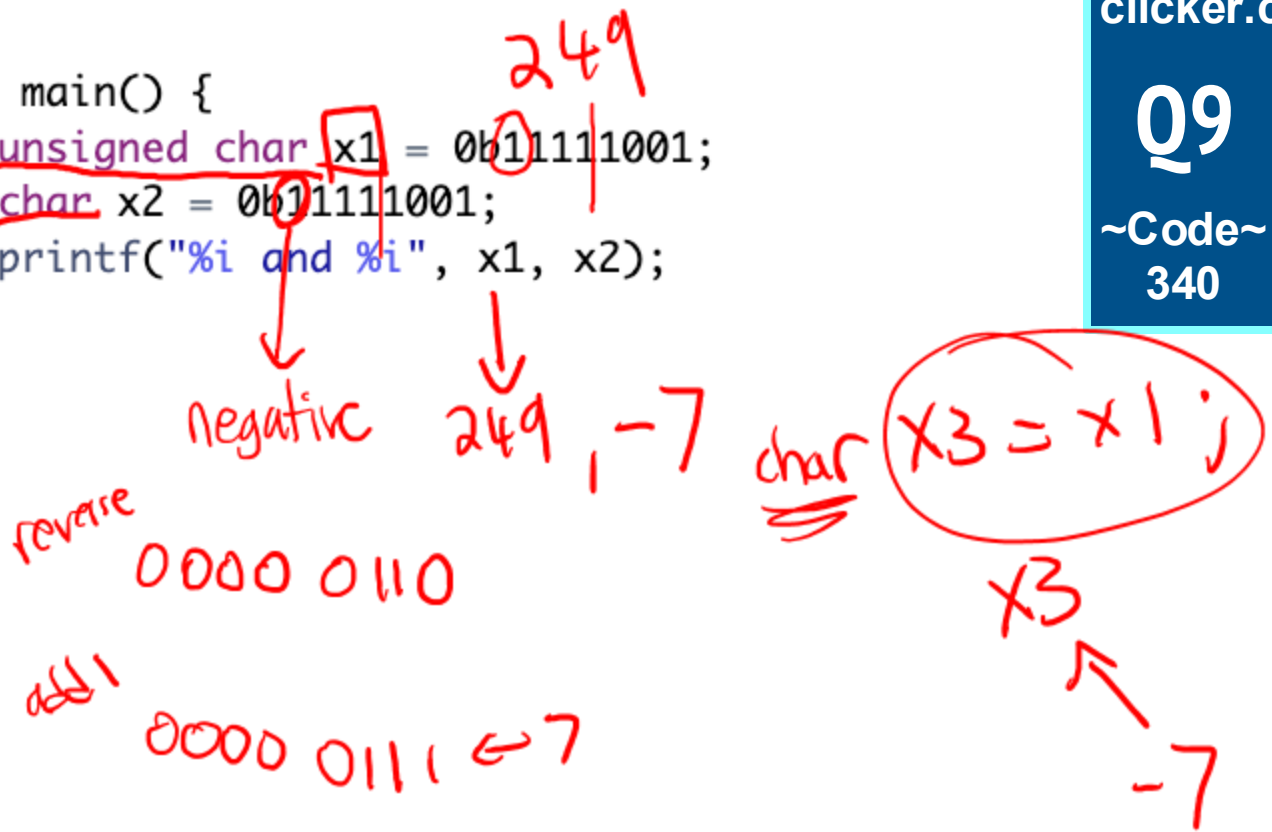
- represent a number ← binary base-2 system
  - character
  - address
- represent negatives ← 2's complement
- metadata + numerical (UTF-8)
- metadata → used, unused

---

bit shifting
bit operations
pointer math
} exam 2 → AND XOR OR

What prints?

```
11 int main() {
12     unsigned char x1 = 0b11111001;
13     char x2 = 0b11111001;
14     printf("%i and %i", x1, x2);
15 }
```



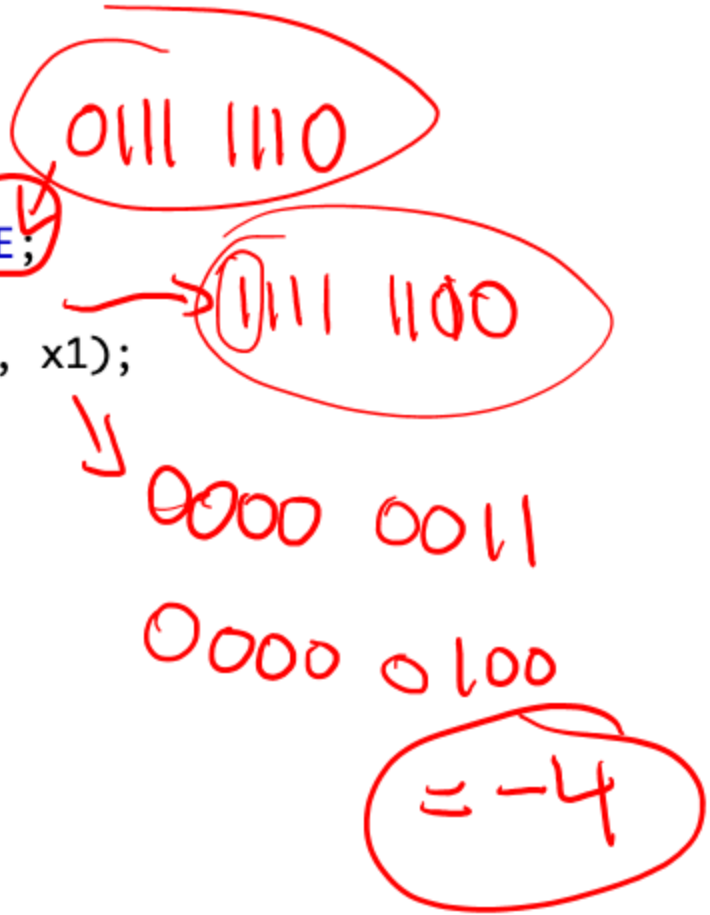
clicker.cs.illinois.edu

Q9

~Code~
340

What prints?

```
11 int main() {
12     char x1 = 0x7E;
13     x1 = x1 << 1;
14     printf("%i\n", x1);
15 }
```



clicker.cs.illinois.edu

Q10

~Code~
340

What code at line 14 would make the program print out a negative value?

clicker.cs.illinois.edu

Q11

~Code~
340

```
11 int main() {
12     int x = 0x07000000;
13
14      //??
15
16     printf("%i\n", x);
17 }
```

Is the following a valid UTF-8 string?

clicker.cs.illinois.edu

Q12

~Code~
340

0000 0001

1111 11

0111

0001 1111 6 bytes

1 byte
4 byte
1 byte

Recall that UTF-8 encodes code points in bytes using the following table:

0xxxxxxx
110xxxxx 10xxxxxx
1110xxxx 10xxxxxx 10xxxxxx
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

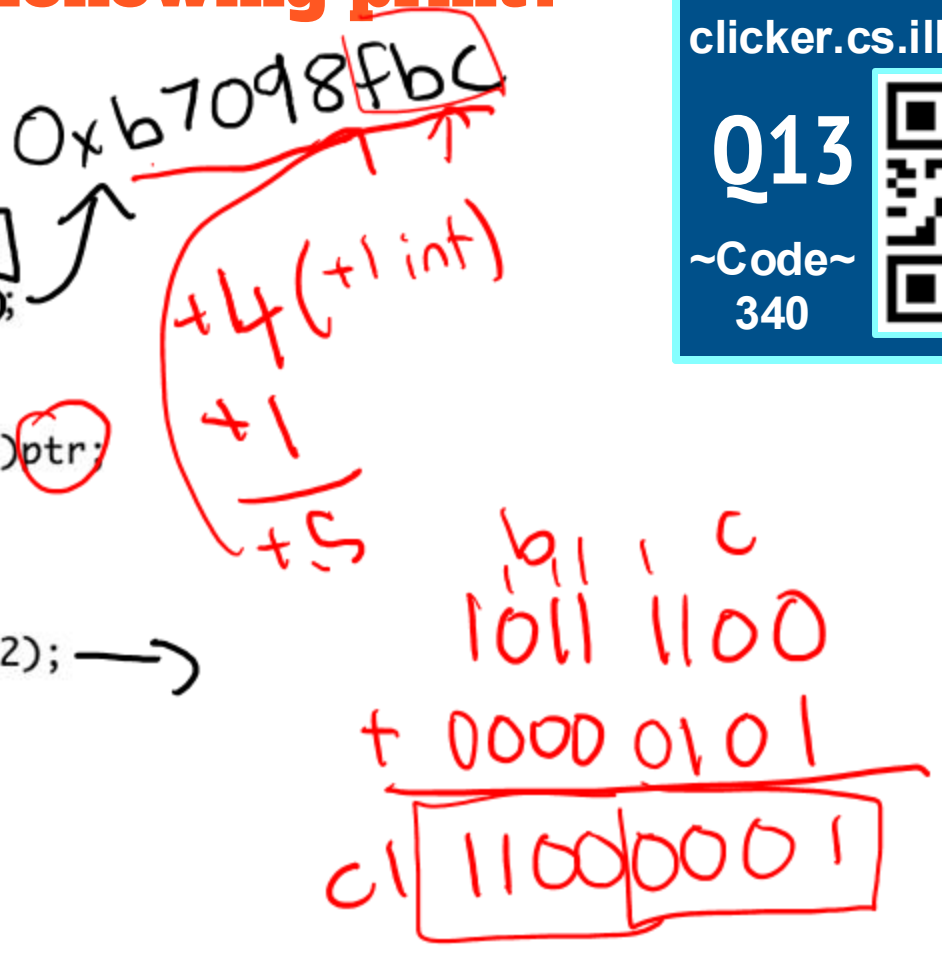
What would the following print?

clicker.cs.illinois.edu

Q13

~Code~
340

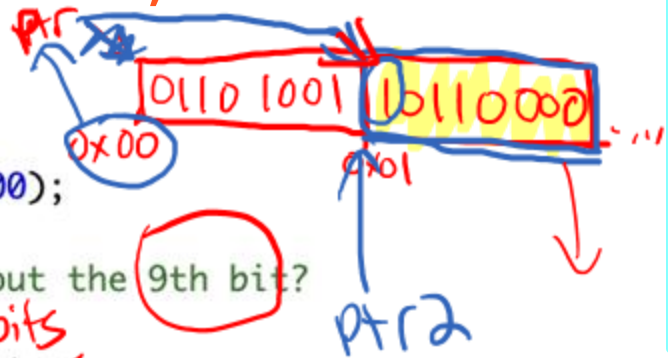
```
11  int main() {
12      int x = 4;
13      //prints 0xb7098fbc
14      printf("%#x\n", &x);
15
16      int *ptr = &x + 1;
17      void *ptr2 = (void*)ptr;
18      ptr2 = ptr2 + 1;
19
20      //prints ??
21      printf("%#x\n", ptr2);
22  }
```



Printing out the 9th bit

```
12  int main() {
13      void *ptr = malloc(500);
14
15      //how would I print out the 9th bit?
16
17
18
19
20
21
22      free(ptr);
23  }
```

Chat with a neighbor, how would you fix this?



clicker.cs.illinois.edu

Q14

~Code~
340

```
12  int main() {
13      void *ptr = malloc(500);
14
15      //how would I print out the 9th bit?
16      ptr = ptr + 1;
17      char *ptr2 = (char*)ptr;
18      unsigned char val = *ptr2;
19      val = val >> 7;
20      printf("%i\n", val);
21
22      free(ptr);
23  }
```


```
main.c: In function 'main':
main.c:22:5: warning: 'free' called on pointer 'ptr' with nonzero offset
   22 |     free(ptr);
      |     ^~~~~~~~
main.c:13:17: note: returned from 'malloc'
   13 |     void *ptr = malloc(500);
      |                 ^~~~~~~~~~
0
munmap_chunk(): invalid pointer
```

This code sometimes prints 1 and sometimes 0, is there a race condition?

threads

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    void *ptr_front = malloc(500);

    //how would I print out the 9th bit?
    void* ptr = ptr_front + 1;
    char *ptr2 = (char*)ptr;
    unsigned char val = *ptr2;
    val = val >> 7;
    printf("%i\n", val);

    free(ptr_front);
}
```

clicker.cs.illinois.edu

Q15

~Code~

340

no

```
14 char str[6] = {'h','e','l','l','o','\n'};
15
16 void *comp(void *mem){
17     char **replace = (char**)mem;
18     char *ptr = str;
19     while(*ptr != '\n'){
20         if(*ptr == replace[0][0]){
21             *ptr = replace[1][0];
22         }
23         ptr++;
24     }
25 }
26 int main() {
27     char* replace0[2] = {"y", "m"};
28     char* replace1[2] = {"o", "0"};
29     char* replace2[2] = {"h", "y"};
30     pthread_t threads[3];
31     pthread_create(&threads[0], NULL, comp, replace0);
32     pthread_create(&threads[1], NULL, comp, replace1);
33     pthread_create(&threads[2], NULL, comp, replace2);
34     for(int i = 0; i < 3; i++){
35         pthread_join(threads[i], NULL);
36     }
37     printf("%s\n", str);
38 }
```

Mutex CV

clicker.cs.illinois.edu

Q16

~Code~ 340

What would this code need to guarantee all replacements are done? AKA prints "mell0"