

CS 340



Part 1 Overview

Agenda

1. Learning Goals
2. CS 340 Part 1 Overview
 - a. Layers of abstraction
 - b. “Hello world” at multiple levels
 - c. Our path forward



CS 340 - Computer Systems




LG: Understanding, at a high level, how we get from electricity to a website.

LG: Developing mental models of how different computer systems work so that you can understand more complex ideas if needed later.

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

MP

340 Part 1 - Overview

Today's LG: To give context to the course material to start seeding concepts and how they fit into the big picture. 

1. Layers of abstraction
2. “Hello world” at multiple levels
3. Our path forward

What we mean by abstraction

I drove to work today high

I left my house, got in my car...

My muscle in my left leg moved
2.3 inches...

↓
low



[Cheryl Harrison](#)

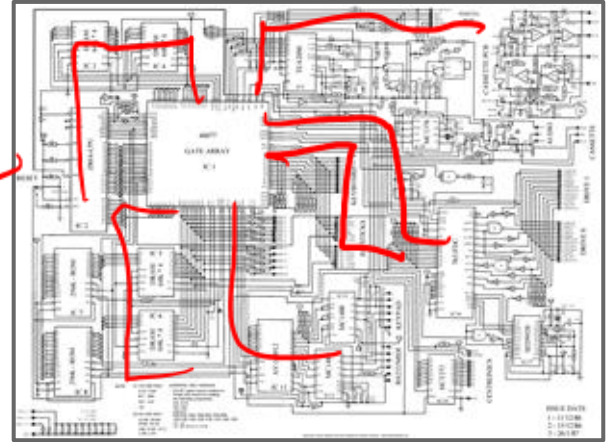
Computer Layers of Abstraction

A screenshot of a C++ IDE window. The title bar shows 'Welcome' and 'main.cpp'. The code editor contains the following C++ code:

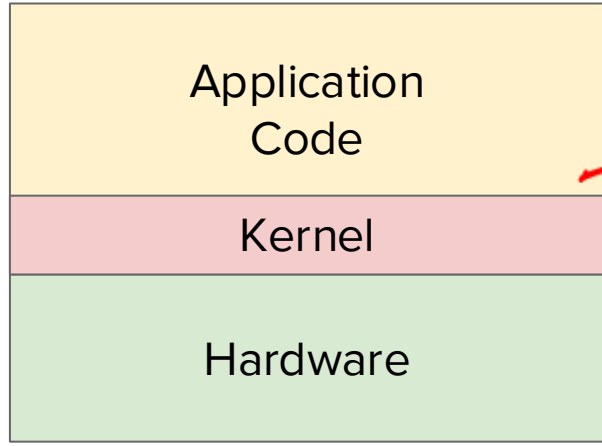
```
1 #include <iostream>
2
3 using namespace std;
4
5 int main(){
6     cout << "Hello World" << endl;
7 }
```

Red annotations include a circle around the 'Run' button (a play icon) in the top toolbar, a red arrow pointing to the output 'Hello World' in the console, and a red arrow pointing to the status bar at the bottom which shows 'Spaces: 4 UTF-8 LF {} C++ Mac'.

340



Big Ideas



→ vs code, compiler

→ code, scheduling, managing resources

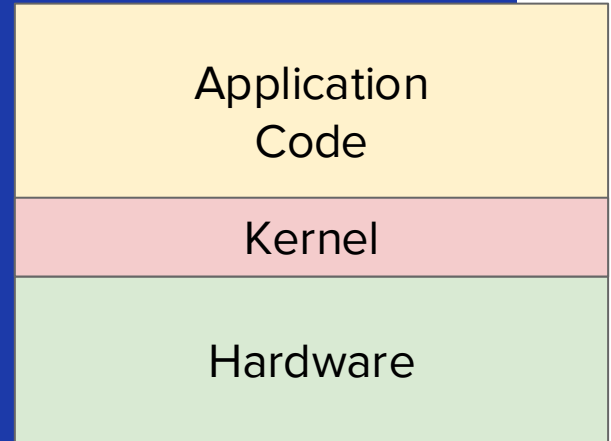
→ transistors, gates, circuits, info storage

What should you remember?

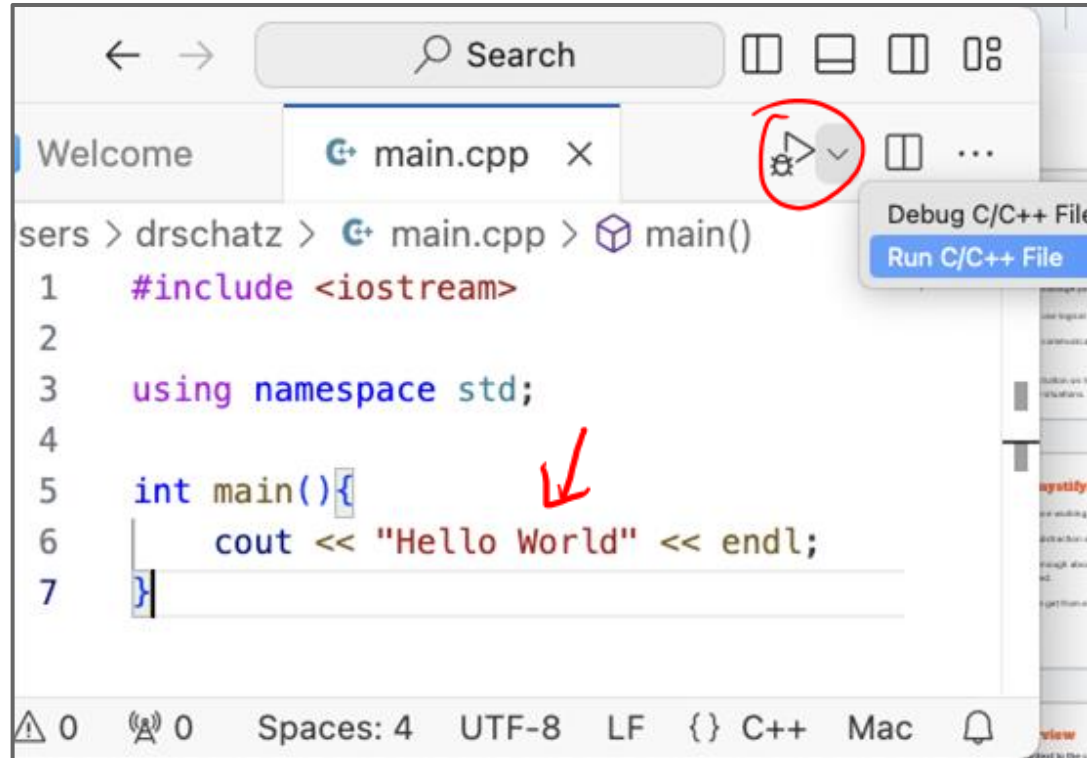
- 340 is about looking into some of the details that have been abstracted away that will help us be customize, debug, and understand computers better.

What should you remember?

- 340 is about looking into some of the details that have been abstracted away that will help us be customize, debug, and understand computers better.
- **This graphic is a simple representation of a computer system.**
 - **Application code** are things you interact with directly.
 - ~~OS~~ ^{kernel} code helps your computer run and manage resources.
 - **All code interacts with the hardware which is what actually executes logic.**



“Hello World” - High Level



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(){
6      cout << "Hello World" << endl;
7  }
```

Debug C/C++ File
Run C/C++ File

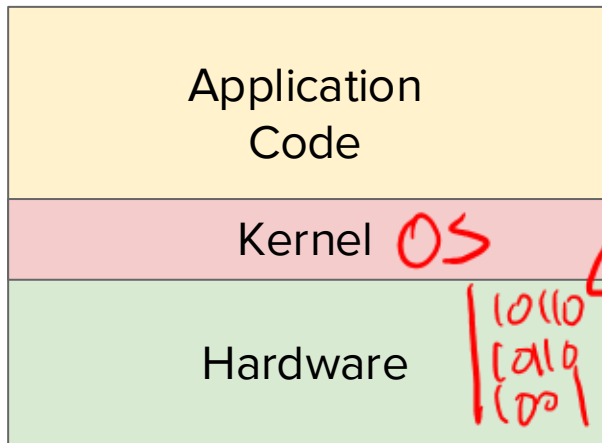
Spaces: 4 UTF-8 LF {} C++ Mac

"Hello World" - Medium Level

```
g++ hello.cc -o exec  
./exec
```

→ Terminal

OS switches to the compiler → OS stores exec



← 1 thing at a time

OS switch run exec

↓ OS switch to terminal

If we remove the operating system can we still execute/run the program?

A) Yes

B) No

C) Unsure



If we delete hello.cc but not exec can we still execute/run the program?

A) Yes

B) No

C) Unsure

clicker.cs.illinois.edu

Q4

~Code~
340



Information in a Computer

Bit - 1, 0
on, off

1

0

Byte - 8 bits

↓

1001 1101

↑

How many bytes are in this sequence of 16 bits?

0000 0110 0000 0111

A) 1

B) 2

C) 4

D) 16

E) Unsure

clicker.cs.illinois.edu

Q5

~Code~
340



What do these bits mean?

↓ ↓
0100 0011

A) 67

B) 'C'

C) Part of a computer instruction

D) Part of an internet packet

E) Can't tell

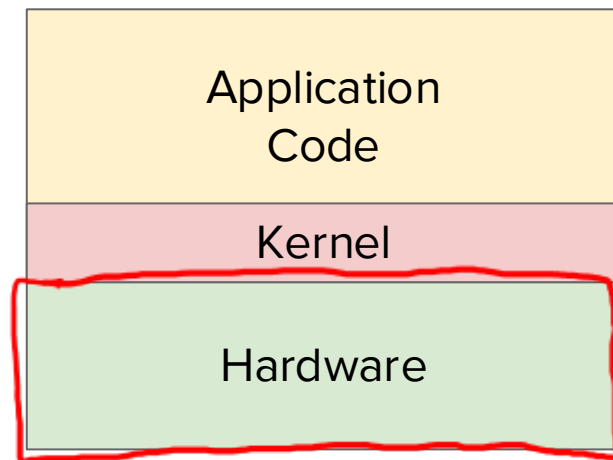
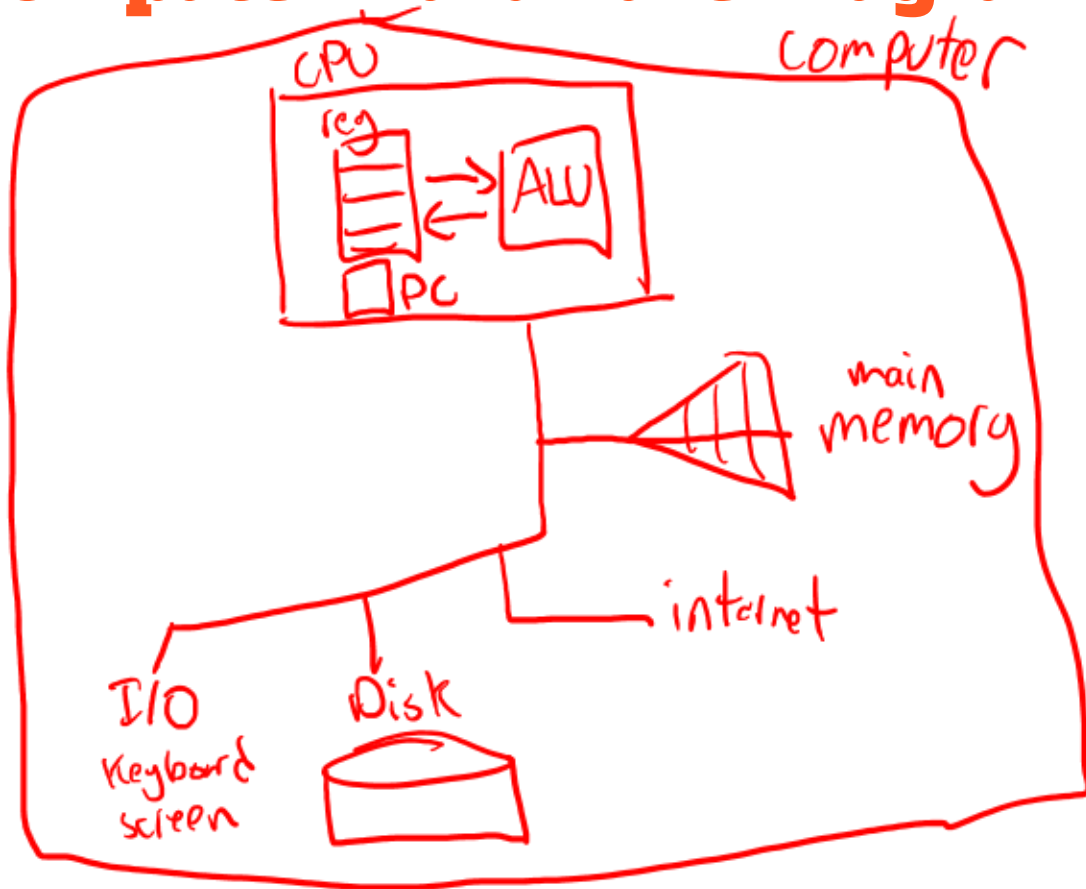
clicker.cs.illinois.edu

Q6

~Code~
340



Computer Hardware Diagram



"Hello World" - Low Level

g++ hello.cc -o exec

`./exec`

I/O → 1,0,0,101 CPU
reg

Super Low Level

Every step involves:

- AND, OR, NOT, XOR: Building blocks of all logic
- Multiplexers
- D Flip-Flops: Storing 1-bit state
- Clock signal: Synchronizing all activity

The Lowest? Level

Every action is controlled by:

- Binary voltage levels (e.g., $0V = 0$, $\sim 5V = 1$)
- A square wave oscillator
- Buses that carry multi-bit voltage patterns across the chip and board

What should you remember?

- There is a lot you don't YET know and a lot of details you may never care to know. Abstraction allows us to often ignore aspects we don't need to dive into.
- All code is run on hardware.
- The following terms exist - Compiler, application, CPU, PC, ALU, OS, memory, disk, I/O.
- The OS has a major role
 - Switches between running programs
 - Interfaces with many parts of the hardware like memory and other devices

Is code executed on hardware or on the OS?

clicker.cs.illinois.edu

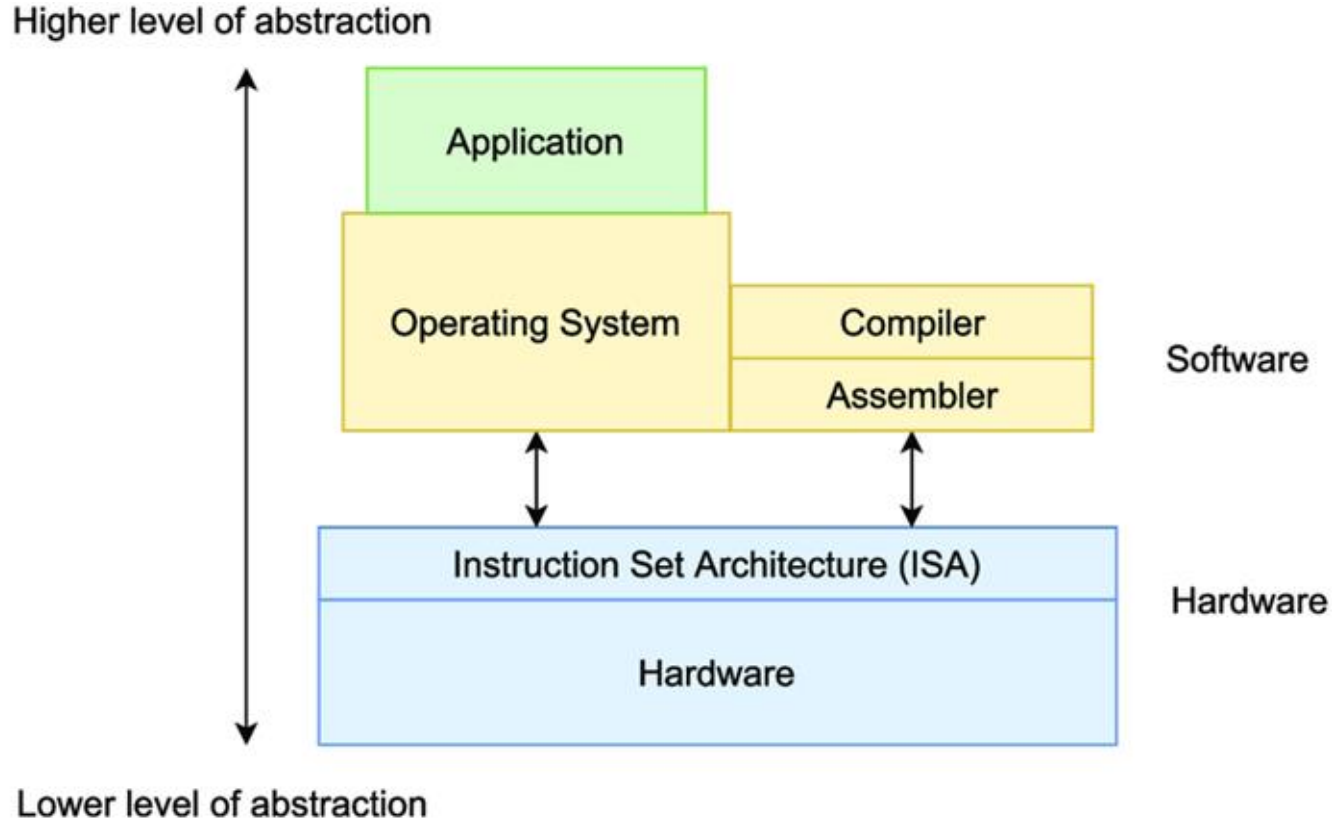
Q7

~Code~
340



- A) Hardware
- B) OS
- C) Unsure

What is misleading about this image?



CS 340 - Computer Systems

LG: Understanding, at a high level, how we get from electricity to a website.

LG: Developing mental models of how different computer systems work so that you can understand more complex ideas if needed later.

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

CS 340 - Part 1 Path

C
gates
binary/hex
memory/caching
endian, bit operations
Processor
v/m
Malloc
OS
Concurrency

2/3

part 2 python



How we will get there?

Lecture - Informational and practical skills

HW - conceptual and smaller tasks around topics learned in class.

MPs - practice working on unstructured coding projects related to systems and topics covered.

Exams - Covers lecture clickers, MPs and HW.