- **HW/Software Interface** is an abstraction layer between physical hardware and high-level instructions (programs) that defines how they talk to each other (instructions, memory interface, etc.).

- Interfaces and higher-level abstractions let us (programmers!) do complicated things with fewer lines of code and without worrying about the actual hardware.

- Hardware likes bit strings (1s and 0s) but humans like simple C or Java style statements.
  - Assembly is the middle ground.
    - C code is *compiled* into *assembly code* (still human readable) and *assembled* into *machine code* (all 0s and 1s).

- This course is all about the interface between the hardware and you (the programmer):
  1. How things are represented (binary encodings) and named (addresses).
  2. How we translate programs (coder-language) to machine code (hardware-language).
  3. How computers keep track of control flow and what (instruction, function, program) happens next, or how function parameters and return values are passed around.

- Representation of numbers is finite in the number of bits we can use to encode it.
  - We lose precision with floats, or range with integers (we’ll cover this more later).

- Memory is important.
  - We only have a limited amount of it: naming and doling it out to programs is important.
  - Writing random data into random locations may not show up for hours or days. Unfortunately, in C, we don’t protect against these: so we’re teaching you how to do it right.
  - Performance is not uniform across various memory locations: we’ll teach you why.

- Learning about the HW/SW Interface is important to make you a better programmer.
  - When things break down, you can figure out why.
  - You can tune your programs to be more efficient on HW and take advantage of memory performance tricks.
  - You can write better and more secure systems software.
  - Sometimes you need assembly to access HW components.