Many debuggers allow you to set the program counter to a specific address then execute one instruction at a time. The debugger provides three stepping commands **Step**, **StepOver** and **StepOut** commands. **Step** is the usual execute one assembly instruction. However, when debugging C we can also execute one line of C. **StepOver** will execute one assembly instruction, unless that instruction is a subroutine call, in which case the debugger will execute the entire subroutine and stop at the instruction following the subroutine call. **StepOut** assumes the execution has already entered a subroutine, and will finish execution of the subroutine and stop at the instruction following the subroutine call.

A **breakpoint** is a mechanism to tag places in our software, which when executed will cause the software to stop. Normally, you can break on any line of your program.

Help

One of the problems with breakpoints is that sometimes we have to observe many breakpoints before the error occurs. One way to deal with this problem is the **conditional breakpoint**. To illustrate the implementation of conditional breakpoints, add a global variable called **`Count`** and initialize it to 32 in the initialization ritual. Add the following conditional breakpoint to the appropriate location in your software. Using the debugger, we set a regular breakpoint at **`bkpt`**. We run the system again (you can change the 32 to match the situation that causes the error.)

```
if(--Count==0){
   bkpt   // place breakpoint here
   Count = 32;
}
```

Notice that the breakpoint occurs on the 32nd time the debugging code is encountered. Any appropriate condition can be substituted. Most modern debuggers allow you to set breakpoints that will trigger on a count. However, this method allows flexibility of letting you choose the exact conditions that cause the break.

anywhere in the world, wherever there is Internet access. EdX's free online MOOCs are interactive and subjects include computer science, public health, and artificial intelligence.