A 64-bit X86 Bootloader From Scratch

Originalartikel

Backup

<html> For most people, you turn on your computer, and it starts the operating system. However, the reality is much more complex as [Thasso] discovered. Even modern x86 chips start in 16-bit real mode and there is a bit of fancy footwork required to shift to modern protected mode with full 64-bit support. Want to see how? [Thasso] shows us the ropes. Nowadays, it is handy to develop such things because you don’t have to use real hardware. An emulator like QEMU will suffice. If you know assembly language, the process is surprisingly simple, although there is a lot of nuance and subtlety. The biggest task is setting up appropriate paging tables to control the memory mapping. In real mode, you have access to the first 64 K of memory unless you use some tricks. But in protected mode, segments define blocks of memory that can be very small or cover the entire address space. These segments define areas of memory even though it is possible to set segments to cover all memory and — sort of — ignore them. You still have to define them for the switch to protected mode. In the bad old days, you had more reason to worry about this if you were writing a DOS Extender or using some tricks to get access to more memory. But still good to know if you are rolling your own operating system. Why do the processors still boot into real mode? Good question. </html>