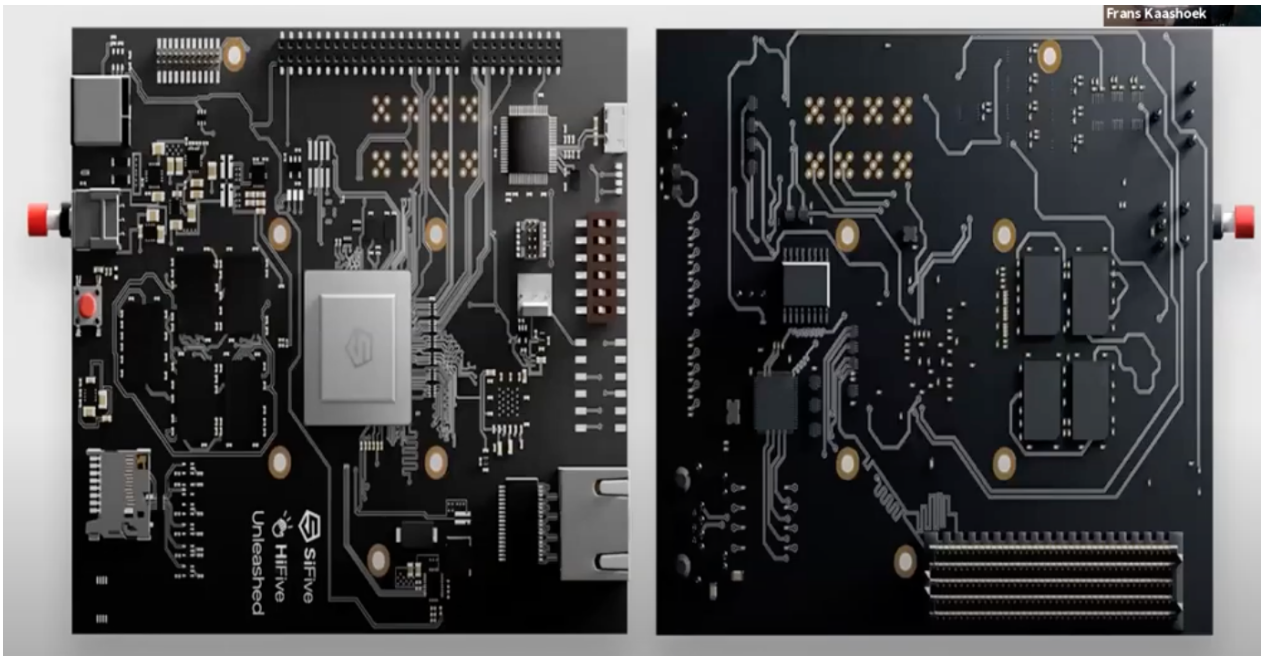
huihongxiao GitBook: [master] one ... Latest commit 84dc16b on Apr 10 History

1 contributor

56 lines (34 sloc) 4.16 KB Raw Blame

3.8 QEMU

QEMU表现的就像一个真正的计算机一样。当你想到QEMU时，你不应该认为它是一个C程序，你应该把它想成是下图，一个真正的主板。



图中是一个我办公室中的RISC-V主板，它可以启动一个XV6。当你通过QEMU来运行你的内核时，你应该认为你的内核是运行在这样一个主板之上。主板有一个开关，一个RISC-V处理器，有支持外设的空间，比如说一个接口是连接网线的，一个是PCI-E插槽，主板上还有一些内存芯片，这是一个你可以在上面编程的物理硬件，而XV6操作系统管理这样一块主板，你在你的脑海中应该有这么一张图。

对于RISC-V，有完整的文档介绍，比如说下图是一个RISC-V的结构图：

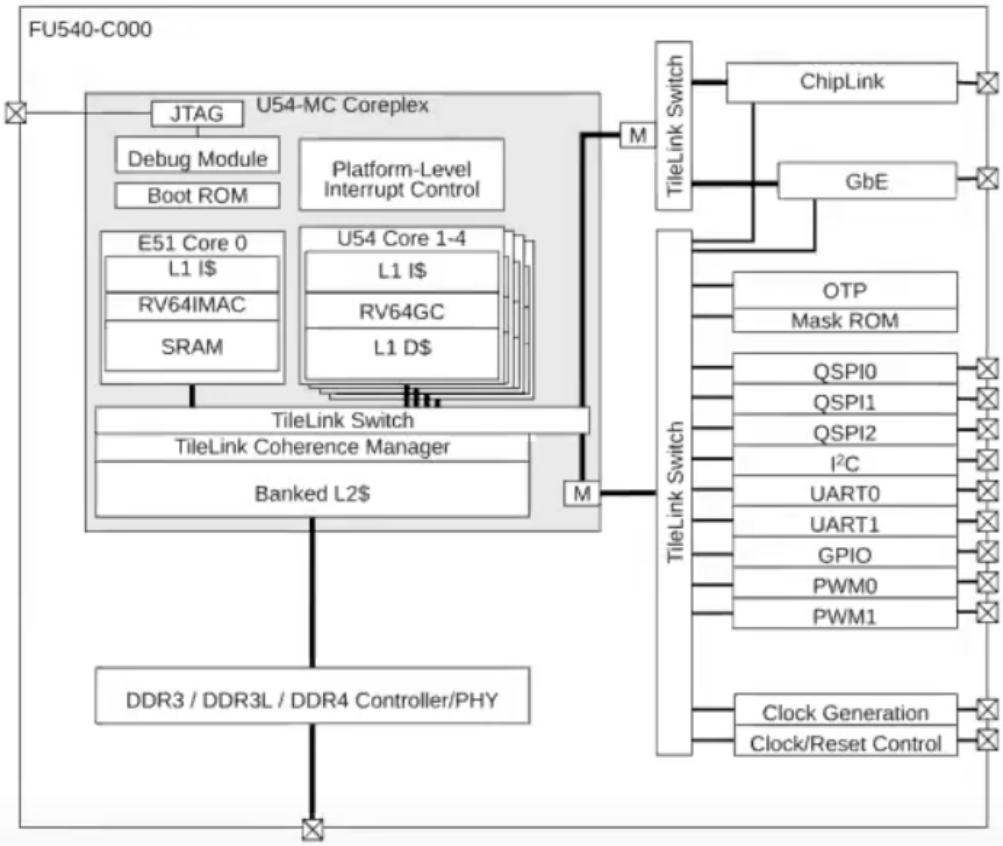


Figure 1: FU540-C000 top-level block diagram.

这个图里面有：

- 4个核：U54 Core 1-4
- L2 cache：Banked L2
- 连接DRAM的连接器：DDR Controller
- 各种连接外部设备的方式，比如说UART0，一端连接了键盘，另一端连接了terminal。
- 以及连接了时钟的接口：Clock Generation

我们后面会讨论更多的细节，但是这里基本上就是RISC-V处理器的所有组件，你通过它与实际的硬件交互。

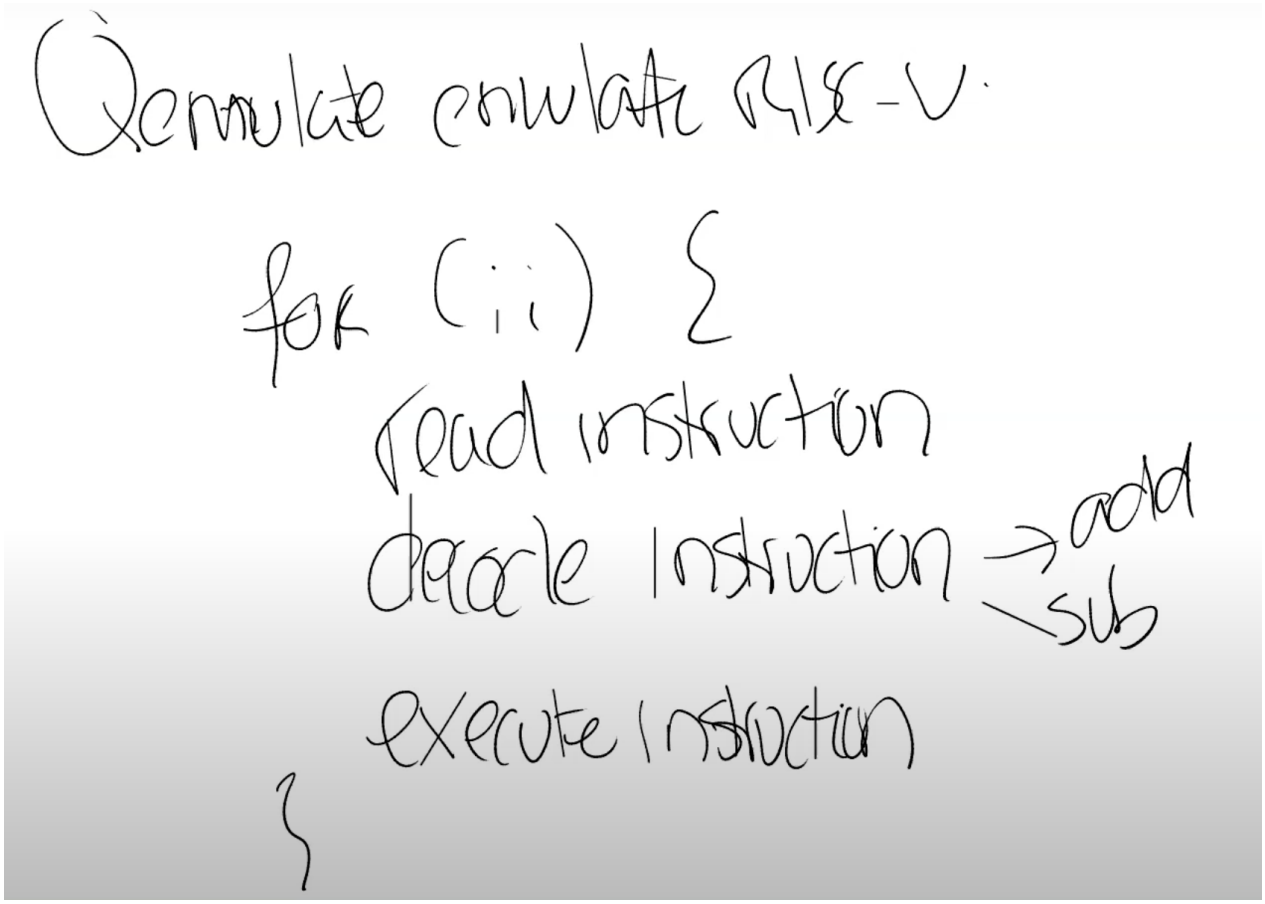
实际上抛开一些细节，通过QEMU模拟的计算机系统或者说计算机主板，与这里由SiFive生产的计算机主板非常相似。本来想给你们展示一下这块主板的，但是我刚刚说过它在我的办公室，而我已经很久没去过办公室了，或许它已经吃了很多灰了。当你们在运行QEMU时，你们需要知道，你们基本上跟在运行硬件是一样的，只是说同样的东西，QEMU在软件中实现了而已。

当我们说QEMU仿真了RISC-V处理器时，背后的含义是什么？

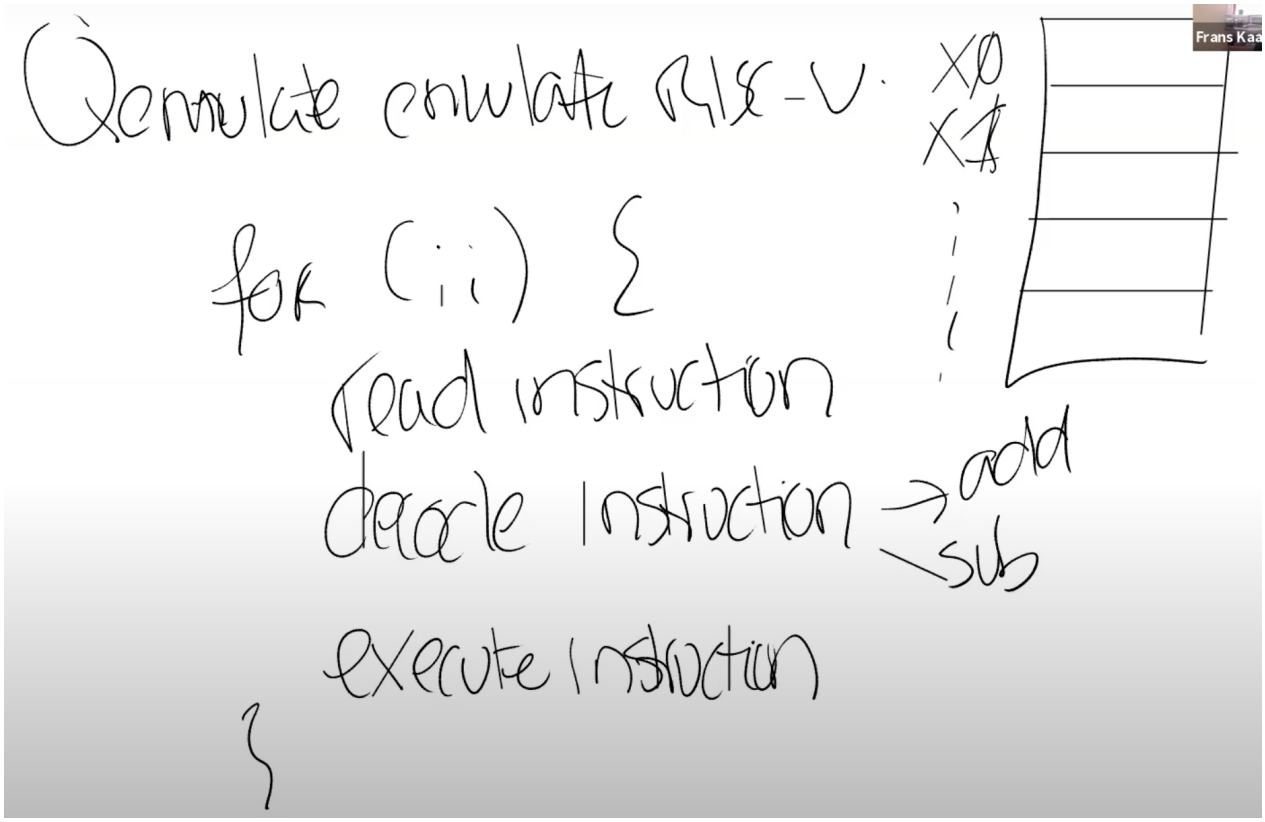
直观来看，QEMU是一个大型的开源C程序，你可以下载或者git clone它。但是在内部，在QEMU的主循环中，只在做一件事情：

- 读取4字节或者8字节的RISC-V指令。
- 解析RISC-V指令，并找出对应的操作码（op code）。我们之前在看kernel.asm的时候，看过一些操作码的二进制版本。通过解析，或许可以知道这是一个ADD指令，或者是一个SUB指令。
- 之后，在软件中执行相应的指令。

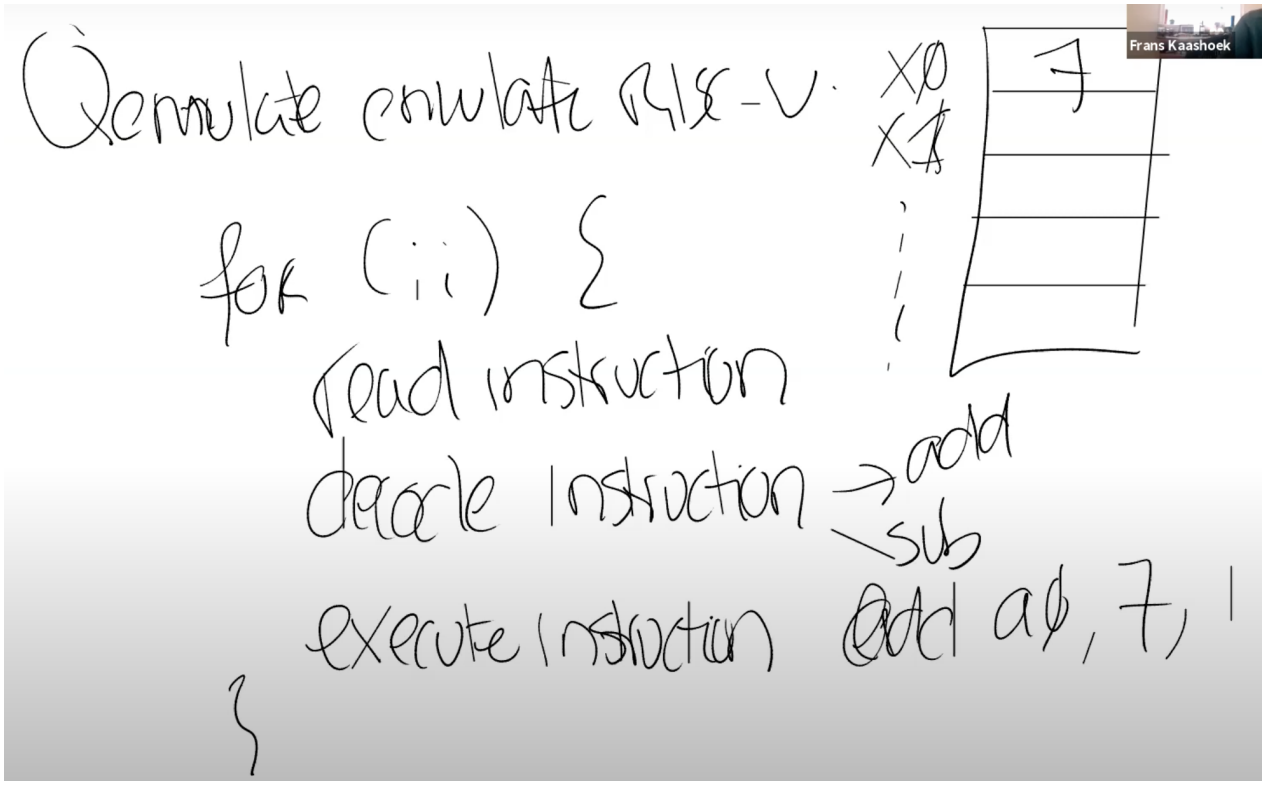
这基本上就是QEMU的全部工作了，对于每个CPU核，QEMU都会运行这么一个循环。



为了完成这里的工作，QEMU的主循环需要维护寄存器的状态。所以QEMU会有以C语言声明的类似于X0，X1寄存器等等。



当QEMU在执行一条指令，比如(ADD a0, 7, 1)，这里会将常量7和1相加，并将结果存储在a0寄存器中，所以在这个例子中，寄存器X0会是7。



之后QEMU会执行下一条指令，并持续不断的执行指令。除了仿真所有的普通权限指令之外，QEMU还会仿真所有的特殊权限指令，这就是QEMU的工作原理。对于你们来说，你们只需要认为你们跑在QEMU上的代码跟跑在一个真正的RISC-V处理器上是一样的，就像你们在6.004这门课程中使用过的RISC-V处理器一样。

这里有什么问题吗?

> 学生提问：我想知道，QEMU有没有什么欺骗硬件的实现，比如说overlapping instruction?
>
> Frans教授：并没有，真正的CPU运行在QEMU的下层。当你运行QEMU时，很有可能你是运行在一个x86处理器上，这个x86处理器本身会做各种处理，比如顺序解析指令。所以QEMU对你来说就是个C语言程序。
>
> 学生提问：那多线程呢？程序能真正跑在4个核上吗？还是只能跑在一个核上？如果能跑在多个核上，那么QEMU是不是有多线程?
>
> Frans教授：我们在Athena上使用的QEMU还有你们下载的QEMU，它们会使用多线程。QEMU在内部通过多线程实现并行处理。所以，当QEMU在仿真4个CPU核的时候，它是并行的模拟这4个核。我们在后面有个实验会演示这里是如何工作的。所以，（当QEMU仿真多个CPU核时）这里真的是在不同的CPU核上并行运算。