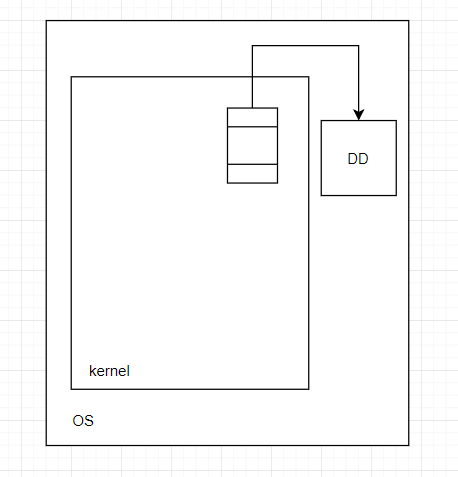
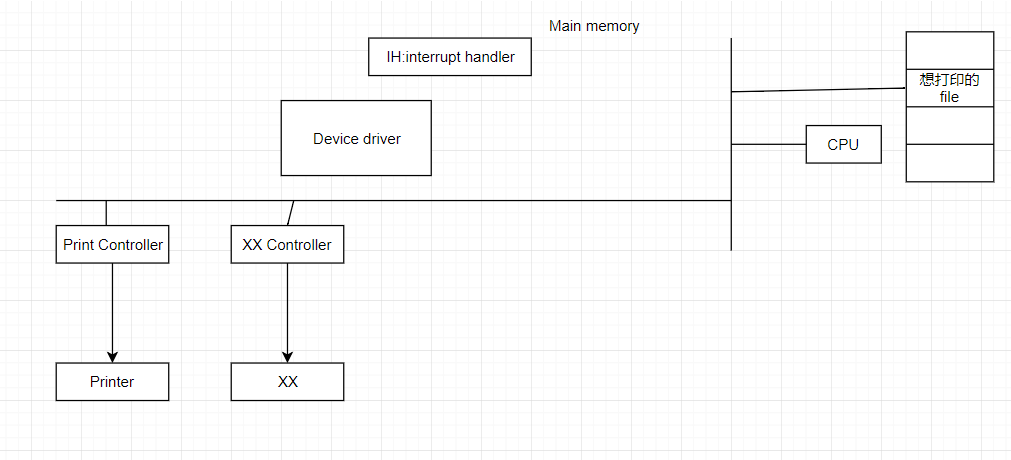
device driver：被认作是OS的一部分

kernel:内核，是OS的核心，DD可以参加修改

kernel有一个特殊table，加入一个DD，table就会加入一个连接DD的LINK



总结：想用一个device，必须装它对应公司的driver, driver是OS的一部分，这样OS仍是最底层的，与硬件接触，控制其他软件//这句话是在讲DD与OS的关系，因为OS就是最底层控制硬件的，如果DD也能控制硬件，不是比OS还底层吗，实际上，他是OS一部分



device driver实际上控制的不是device，而是对应的controller

打印：1.poll:device driver 查看controller，问他是不是busy.如果是，done（就不打印）

2. transfer.假如file是640M。 buffer memory是64M，那么就要transfer十次

3.poll：不停检查controller，直到done（实际不同）

poll会浪费大量时间，因为OS必须要找device driver问他有空没、

transfer会浪费时间也是因为要用device driver来分割，转移file到buffer memory

运行dd的时候，他的operating system什么都不做

3/ interrupt 打扰

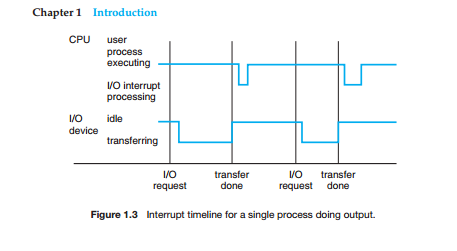
如果打印这个操作做完了，第三步不使用Poll而是interrupt，打扰

当device controller结束了打印，发出interrupt指令，当CPU的interrupt register收到interrupt请求，CPU暂停当前任务，并且激活interrupt handler, IH

Interrupt handler也是OS的一部分，他会check每一个device，看看是哪个device controller发出的interrupt指令

然后CPU会执行device driver，改成done

IO interrupt

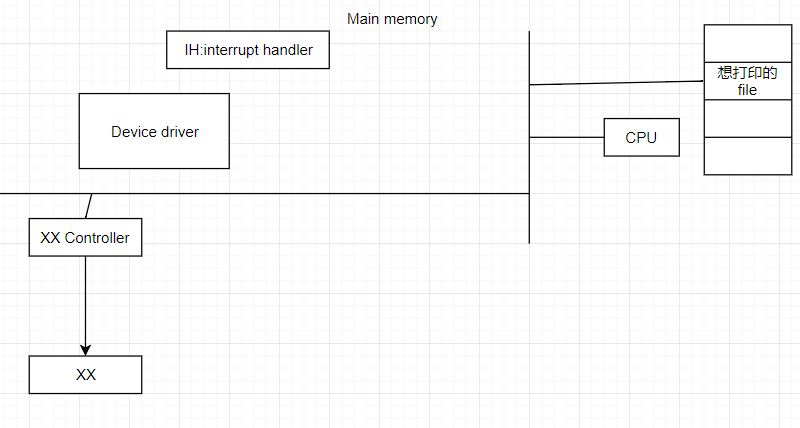


1.CPU发出打印指令，经过一段热身，printer开始由idle，变成work，结束后发出interrupt，经过一小段等待，CPU停止当前任务，处理interrupt

因为打印机运行时，CPU仍然在不停处理其他任务，所以很快

controller内部有一个小小处理器，他只需要知道从main memory哪里开始，要transfer多大容量

会和CPU产生冲突，

CPU正在处理一个program .

controller processor 在CPU一个CYCLE的休息时期间，偷偷使用data bus， CPU一使用就还回去

坏处：可能printer移动数据变慢

好处，假设这个program用时92S，因为controller steal了data bus，可能需要95s,然而如果不steal的话，可能需要92s+打印的时间//5分钟

所以总的过程是

1.POLL

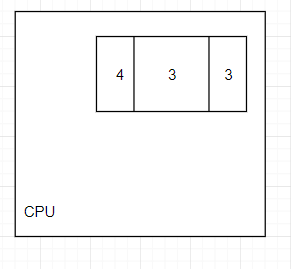
2.TRANSFER：弃用DD-DEVICE DRIVER/使用DMA direct memory access偷取

3.弃用POLL,使用interrupt

第一步为什么用Poll而不是interrupt，因为假设不同的device都不停的interrupt “I am free”，占用资源而且我们并不在乎，我们只又要用的时候poll,来看他是free还是busy

第二步用DMA，这个device趁CPU空闲时间，偷偷地把file从memory转给buffer register

第三步，任务完成，interrupt，告诉CPU我任务完成了



CPU的内部设置了多位digit，每个digit控制多个device，一个device发出interrupt，找对应的digit。

如果这几个device同时运行，只能一个个interrupt

总结：直到 IO INTERRUPT /IO POLLING/DMA