Input-output organization

I/O设备比如说打印机，鼠标键盘

CPU和IO设备的数据传输是通过一个或多个BUS进行

如果一个CPU有多个BUS，那么一个和MEMORY相连，一个和IO相连

如果只有一个BUS，那么就叫做SYSTEM BUS

IO设备通过一个叫INTERFACE接口的设备和BUS相连

一个INTERFACE要满足的有：

1：使计算机知道IO设备的状况（比如说IO设备是否做好准备来接受一个新的命令）

2：对IO对电脑的输入的信息有一个暂时的存储

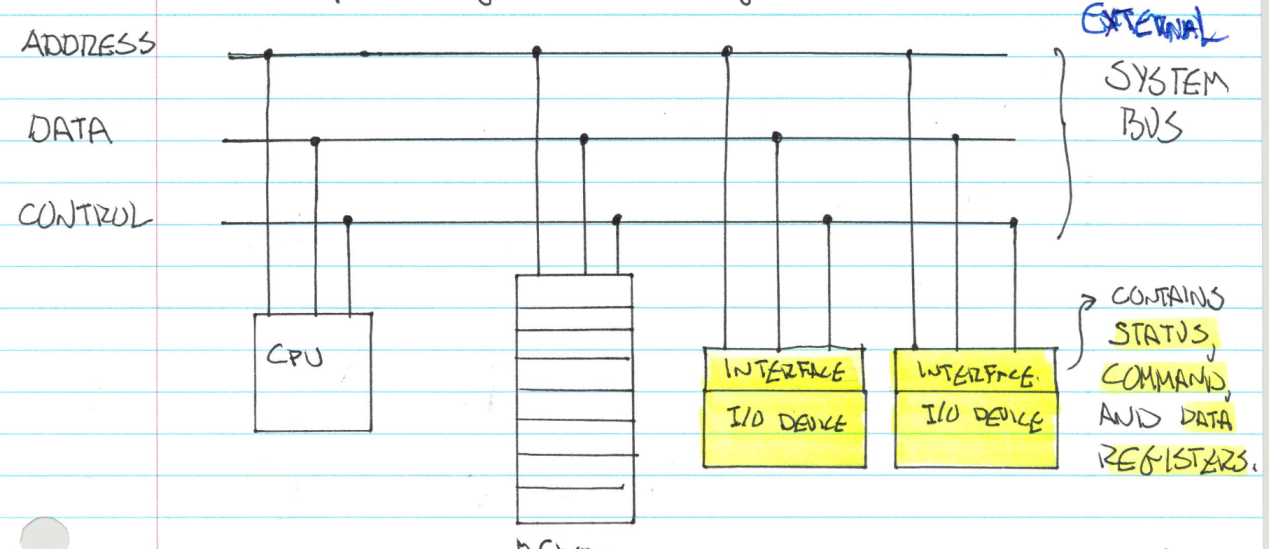
3:把电脑的命令传给IO

4：对电脑对IO的输出的信息有一个暂时的存储

5：告知计算机当一些设备操作完成时

6：告诉计算机特殊情况

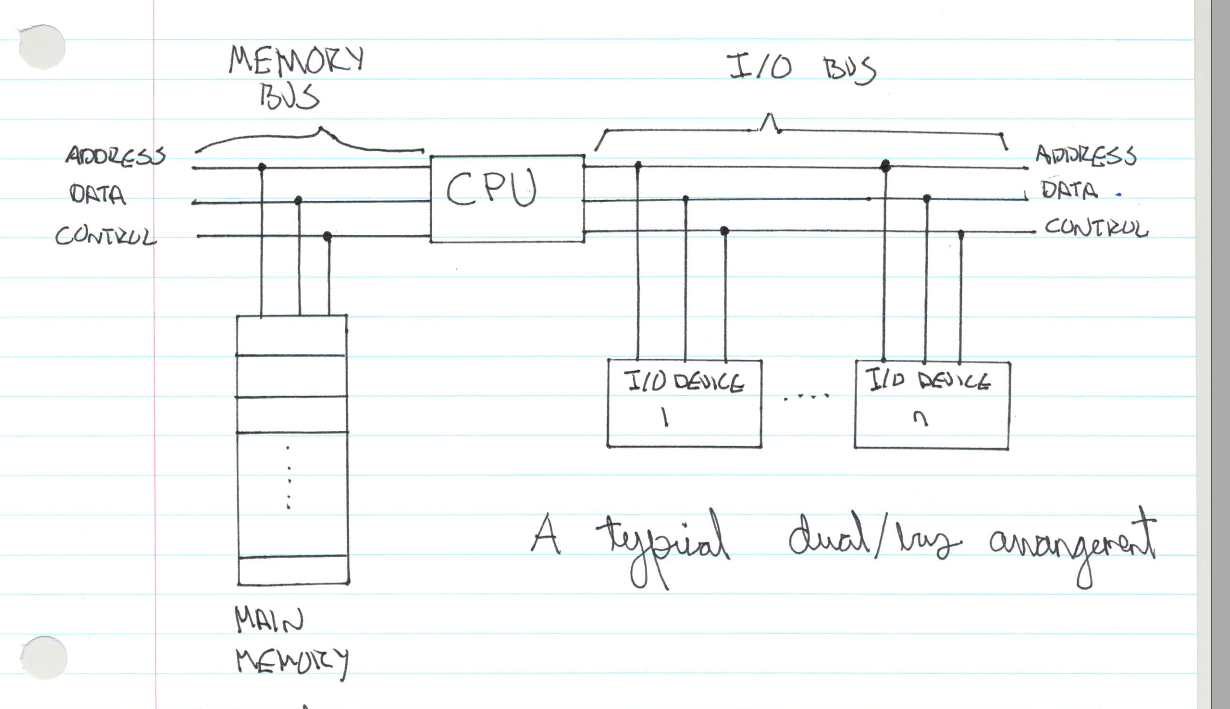
一个标准的SINGLE-BUS 结构



INTERFACE 包含了 状况，指令，DATA 寄存器

一个multiple bus结构

有一个memory bus有一个IO BUS，每一个设备有一个独立的ADDRESS，CPU通过把ADDRESS放到ADDRESS LINE上来选取设备



DATA LINE当和IO设备相连时，传递的信息可以是DATA,DEVICE STATUS设备状况和 COMMAND指令和INTERRUPT INFORMATION中断信息

ADDRESS Line 传递 Interface 里的ADDRESS

CONTROL DEVICE:传递 CPU 和 DEVICE之间的 control information ,这些信息包括

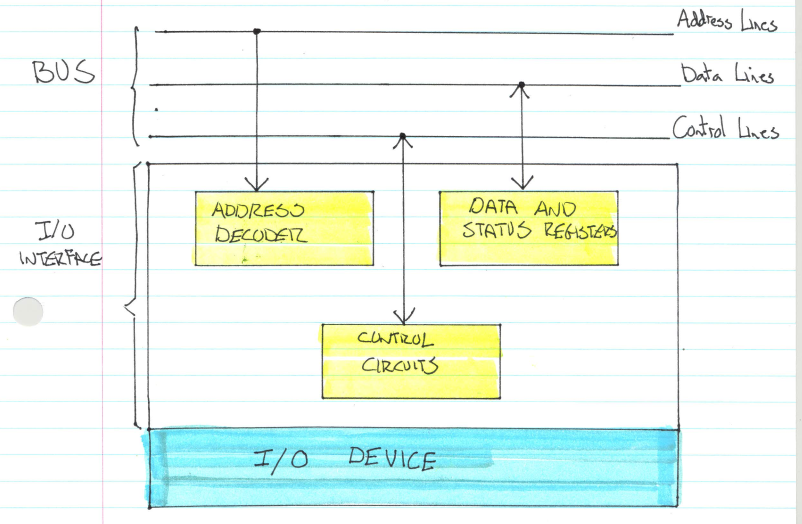
1.对于使用BUS的请求，每一次IO设备需要使用BUS，他们必须请求进入BUS权力

2.grant for bus usage,bus使用的授予，CPU制定好的对于各个DEVICE先后使用B US的授予

3.INTERRUPT SIGNAL：中断信号

4.TIMING SIGNAL：时间信号

INTERFACE结构图



ADDRESS DECODER:可以让设备地址传到address line上

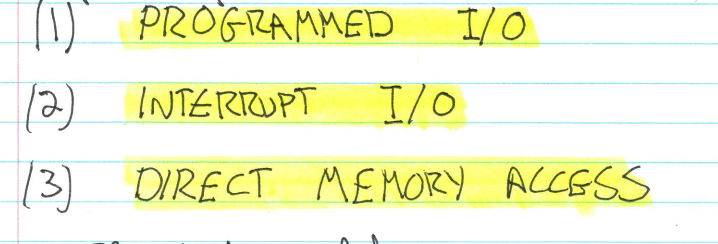
DATA REGISTER,保存CPU 和 REGISTER之间传递的数据

STATUS REGISTER，保存设备的状态信息

Input/output techniques技术

IO设备应用三种技术来传递或接受数据

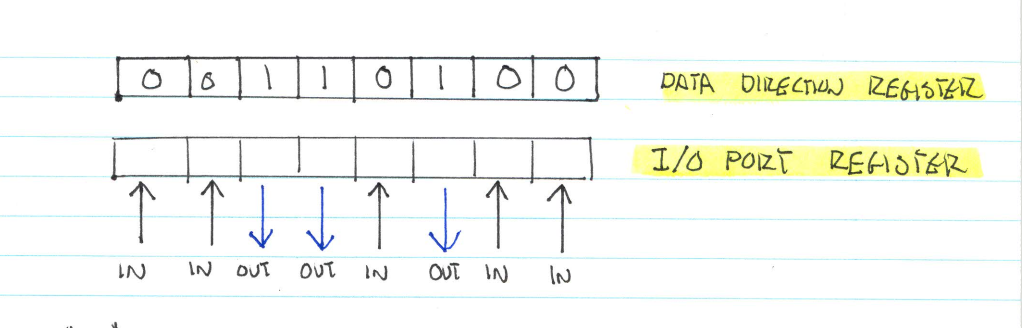
Port端口



第一种方法用了一个程序来连接外部的设备通过一个叫做 INPUT/OUTPUT port的寄存器，一次传输1BYTE的DATA0 1byte=8bit

IO PORT是早期设计的interface的一部分

IO PORT是输入还是输出可以通过另外一个特殊寄存器叫做COMMAND REGISTER OR DIRECTION REGISTER



DATA =0,IN,DATA=1,OUT IO PORT里面另有数据

STANDARD IO

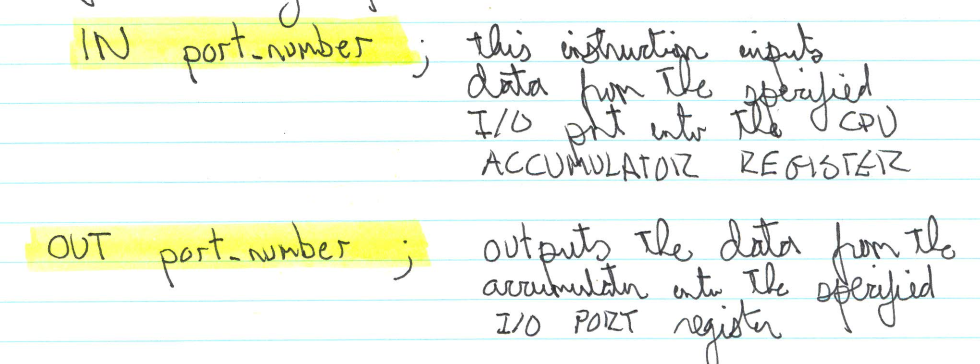
用了两种特殊的CPU指令 IN 和 OUT和一个特殊的CPU CONTROL LINE叫做 IO/M

这个LINE是1代表着 IO操作正在进行，0代表着数据传输。IN和OUT的时候是1（IO的操作）,0的时候是load or store（memory的操作）

Memory mapped IO

CPU不区分 IO设备和memory的地址区别，因此，没有 IO/M线。 CPU在memory address 分配一部分来作为 IO PORT，普通的 memory access操作用来进行对这些memory-mapped PORT进行读写操作

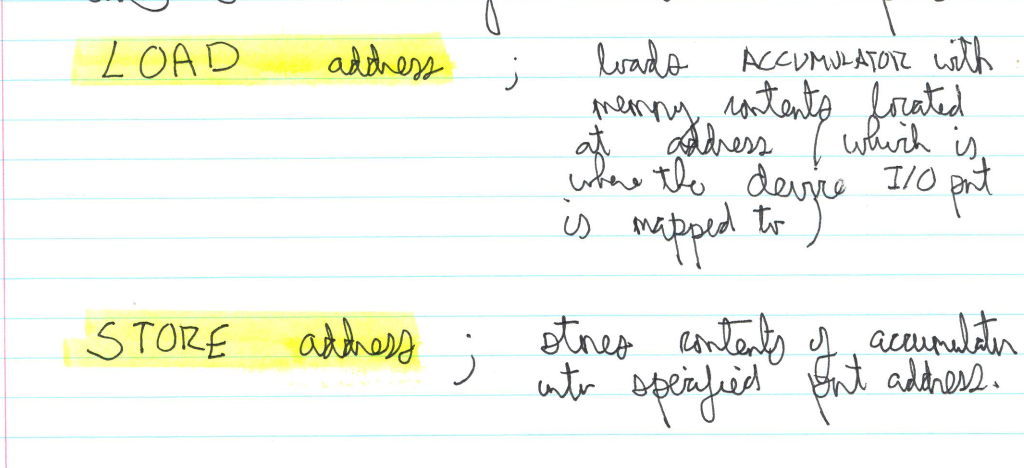
Standard io



IN PORT NUMBER,把特定IO PORT里的数据输入到 CPU的 accumulator register中

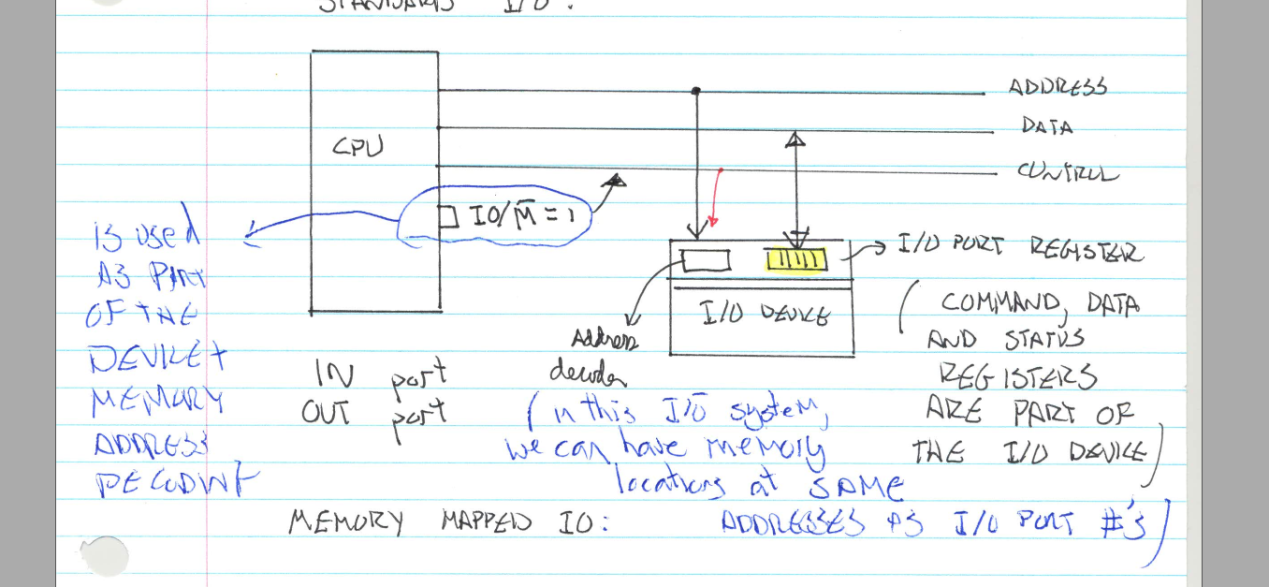
OUT PORT NUMBER ，accumulator register的数据被输出到特定的 IO PORT register中

MEMORY IO

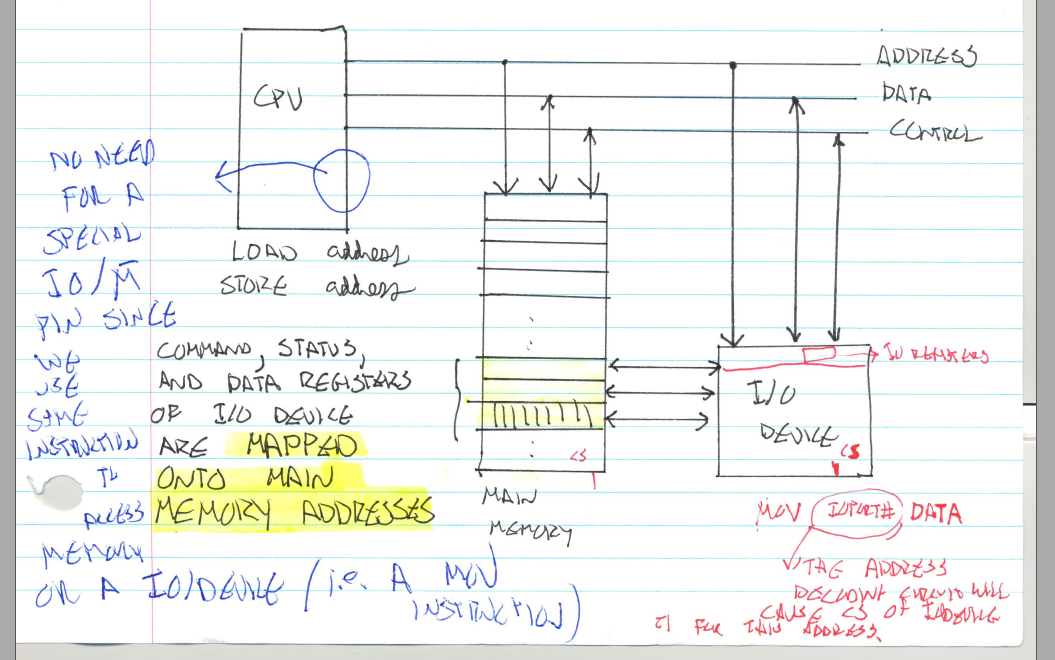


LOAD ADDRESS,特定地址的 IO被写进accumulator里

STORE，accumulator的内容被存进特定的 port address里



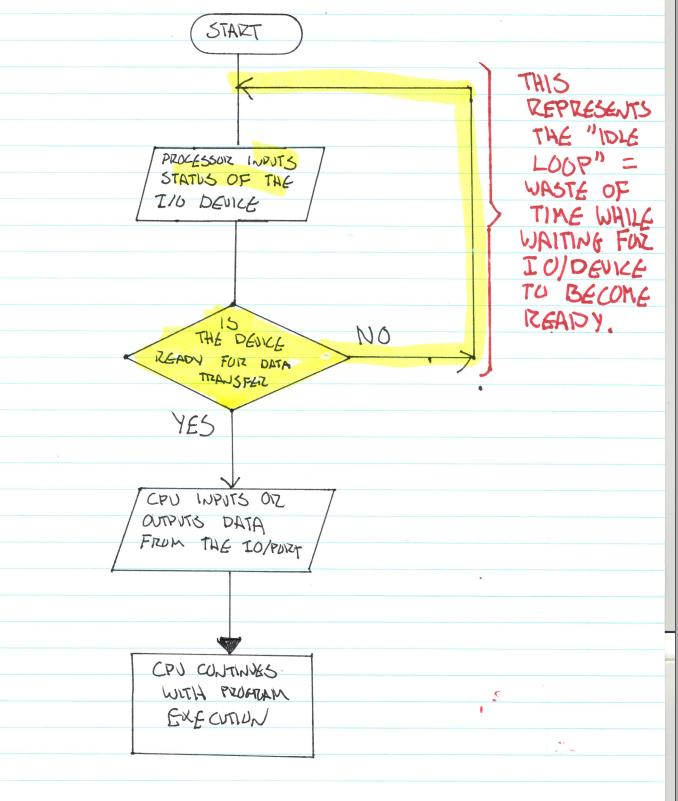
IO/M=1，所以进行右边的操作，这个系统中 IO DEVICE可以和 MEMORY具有同样的ADDRESS



Memory mapped

没有特殊的io/M因为用的一条线，IO DEVICE的命令，状况和DATA寄存器都被MAP到 memory address里

一个标准的programmed io 流程表

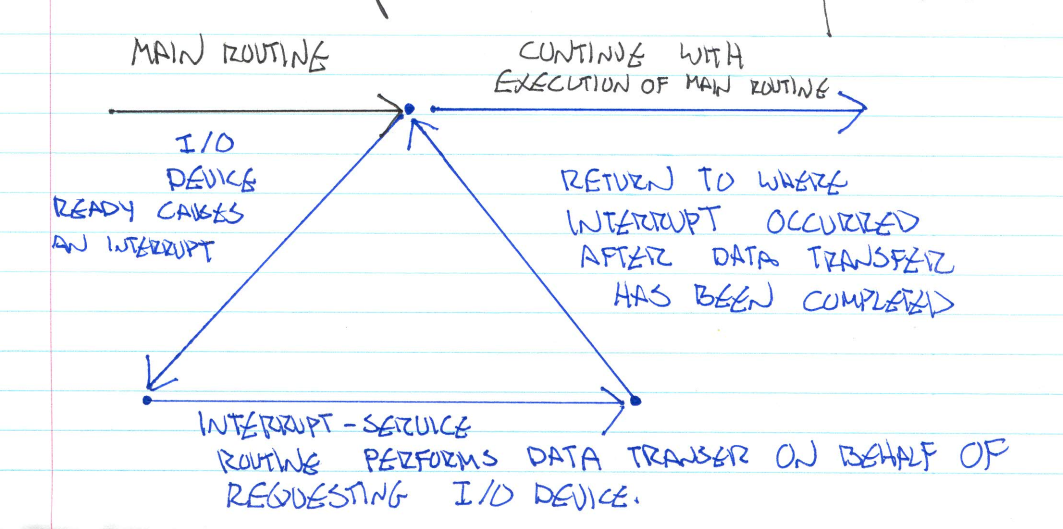
CPU接受IO的状况，如果设备没准备好传输信息，那么无限循环，大多数时间都浪费在这，准备好传输信息，CPU INPUT OR IO OUTPUT

CPU 继续运转

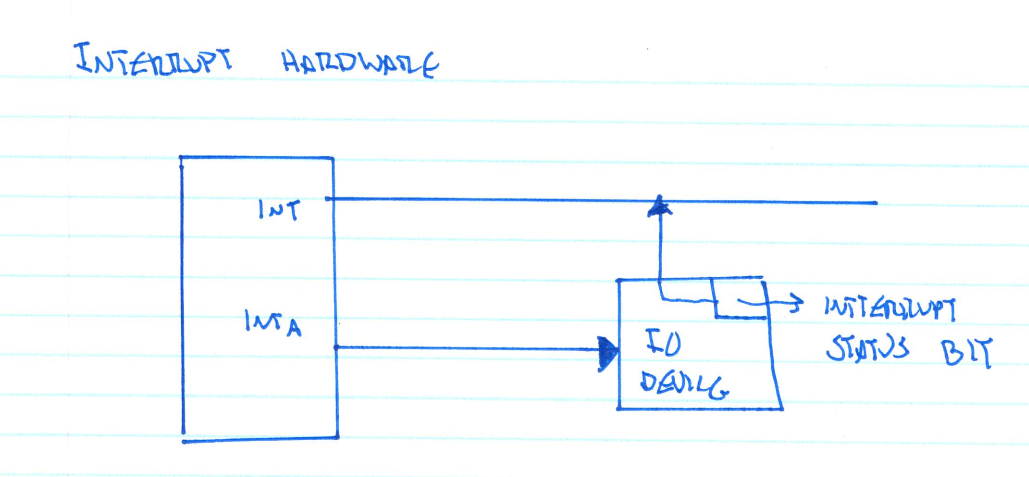
Interrupt IO IO的中断

Programmed io 有个巨他妈大的缺点就是 总是在 做上图那个IDLE LOOP,浪费CPU机能， interrupt IO消除了 这个等待过程

当CPU进行一个 Main routine主程序的时候，而IO设备已经准备好传输1BYTE的数据,IO设备给CPU传输一个INTERRUPT信号，那么CPU就会放弃主程序的进行并进行一个interrupt secure routine中断安全程序,这个程序请求IO设备执行数据传输，数据传输结束后回到原来main routine中断的地方然后继续



INTERRUPT HARDWARE



IO设备通过接入INT线来申请INTERRUPT,CPU进行如下操作来应对INTERRUPT

1.保存现有内容的PC

2.把INTERRUPT SERVICE ROUTINE的开始地址保存到PC里

3.CPU告诉INTERRUPT DEVICE他的INTERRUPT请求已经被识别了，是时候停止INTERRUPT SIGNAL 了

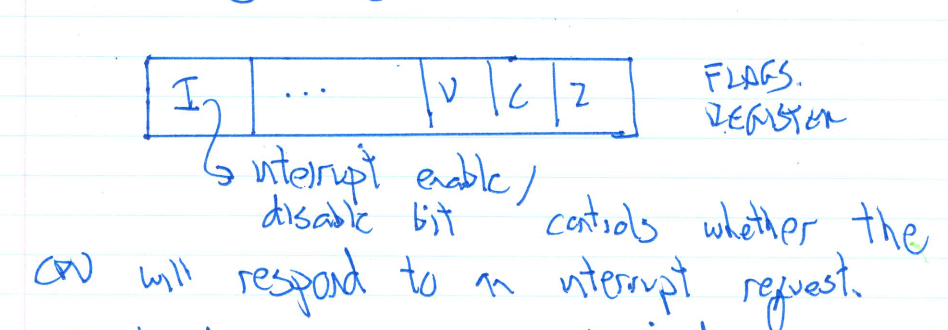
为了完成3,有两个主要步骤，1接入线路INTA，当CPU从INTA传出信号时，他就取消INTERRUPT请求

2或者在IO内部设置一个硬件设备，这样IO设备进行一个操作会打断INTERRUPT SERVICE ROUTINE。

Enabling/disabling interrupt

如果有一个以上的设备申请interrupt，就有可能进行一个device申请的时候，另外一个device申请interrupt

1. cpu hardware放一个特殊的bit在CPU FLAG REGISTER里



I: interrupt enable/disable bit，控制CPU是否回应INTERRUPT请求

这个hardware是被这样设计的：当int line被接入时，CPU disable这个BIT,然后CPU保存当前PC并LOAD ISR的Pc地址。

ISR的最后一个操作就是重新存下那个PC然后重新enable那个BIT

这样建立在硬件上的Interrupt无法应对多重设备

建立在几乎差不多的软件的Interrupt也同样无法应对

Processor的硬件设备必须被如此设置：

1任何Pending(未发生的)interrupt请求都被忽略直到完成执行ISR之后

2RTE instructor必须要完整

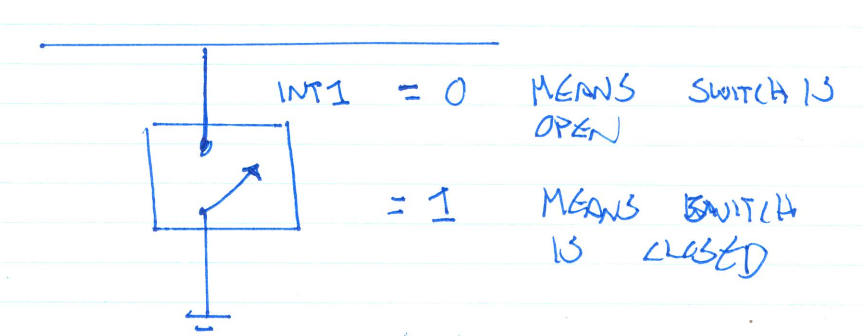
Open-drain common interrupt requesting

叫做 open-drain 的特殊逻辑电路允许多个输出驱动在同一个BUS上

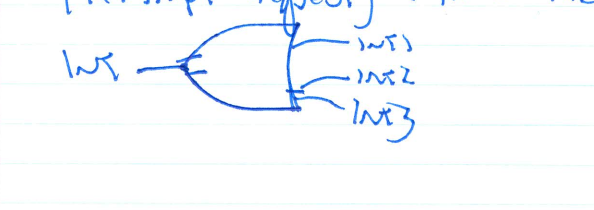


Non-interrupt请求就是每个开关都是开，没有电流接地，INT电压是HIGH，然后CPU内部还有个否门，CPU内部是0

当一个或多个想interrupt时候，关掉开关，电压变成0，CPU内部是1



开关打开算0，关上算1

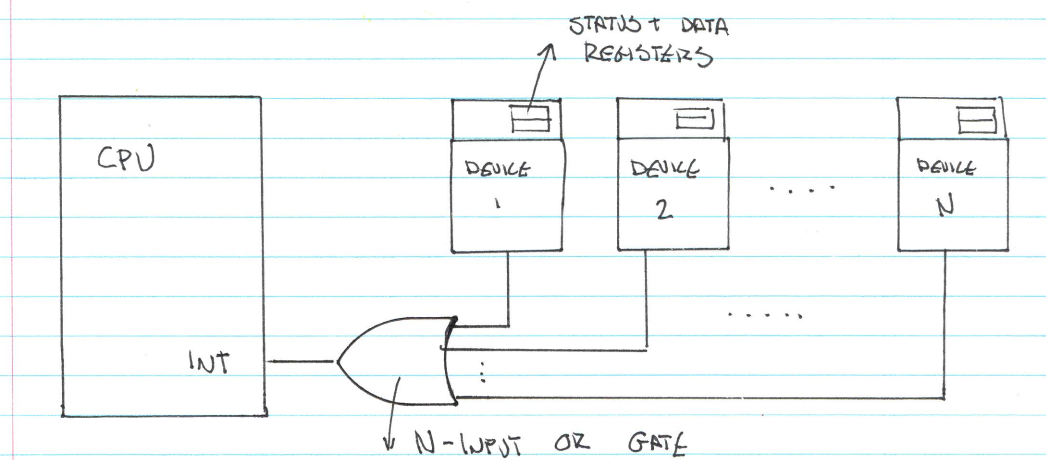


实际关系是1个或以上关上，INT就是1，全部打开，INT是0

一个return 操作，把中断的地方Pop给PC，然后重新运转

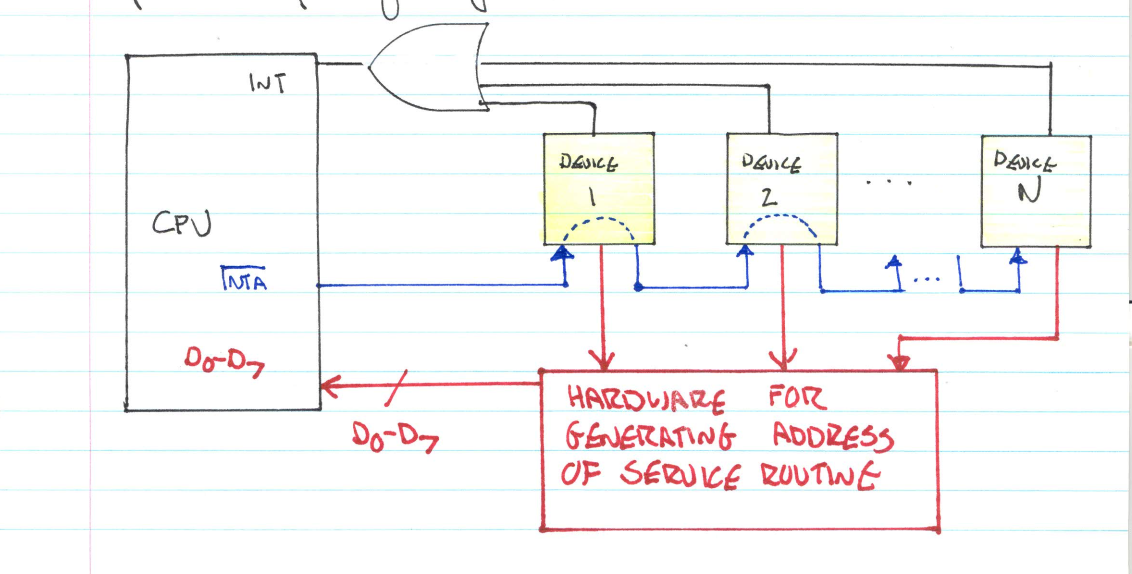
COMMON INTERRUPT LINE

如果许多DEVICE被连在一根COMMON INTGERRUPT LINE上，那么CPU必须由一些装置来识别到底是哪个DEVICE发出INTERRUPT请求，总共有两种方法:POLLING和DAISY CHAINING



当一个或多个DEVICE激活了INT LINE时，CPU内有个事先编好的POLLING程序，他会依次检查每一个DEVICE的STATUS，第一个状况为ready的设备将会被负责，这个方法是软件负责的，因此很慢

DAISY CHAIN INTERRUPT



CPU完成现有指令，把 PC PUSH到STACK上

接着CPU通过INTA发出INTERRUPT承认信号到优先级最高的DEVICE（DEVIICE1）

如果这个DEVICE发出INT信号（申请INTERRUPT），那么他就会接受INTA的信号，否则顺延至下一个

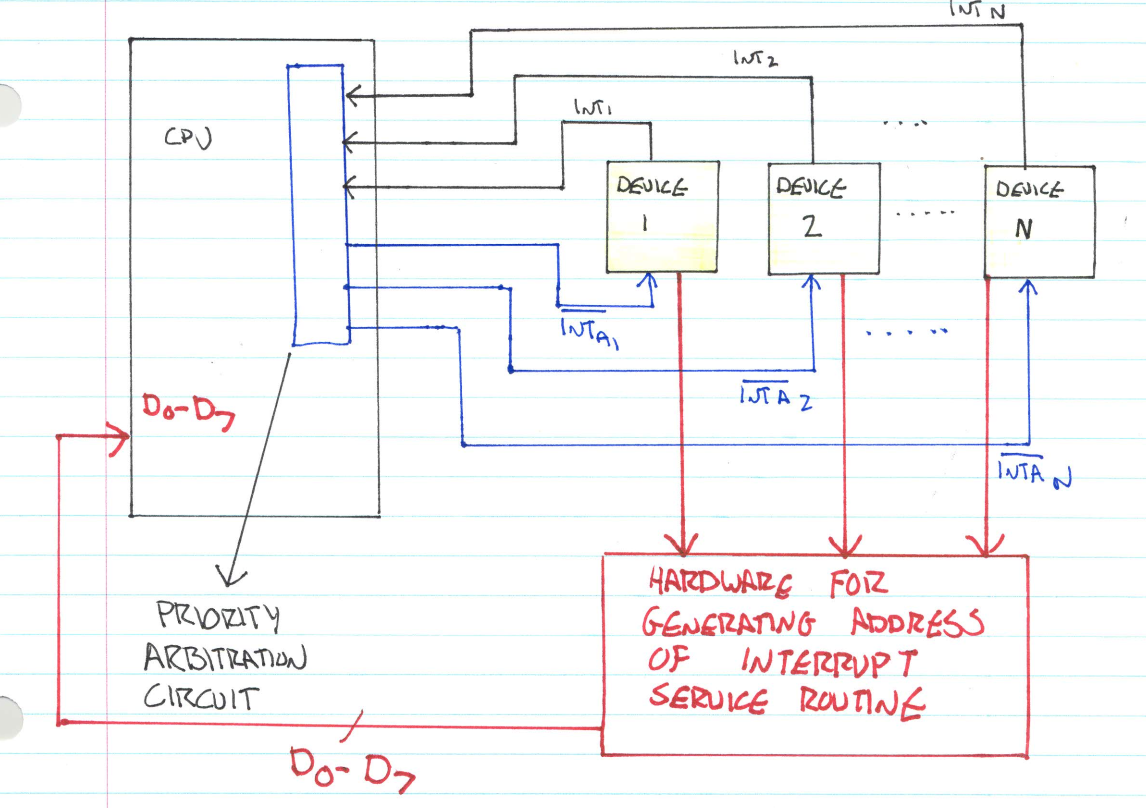
一旦接受了信号,DEVICE就把他ISR的地址给Hardware,然后通过DATA BUSS传回CPU

这种方式更快

MULTIPLE-INTERRUPT HARDWARE

如果CPU收到了多个interrupt，那么就会解码哪个Interrupt先处理，哪个延后，这个可以简单的被implement（执行）在priority电路里通过赋予不同的interrupt line不同的优先级。

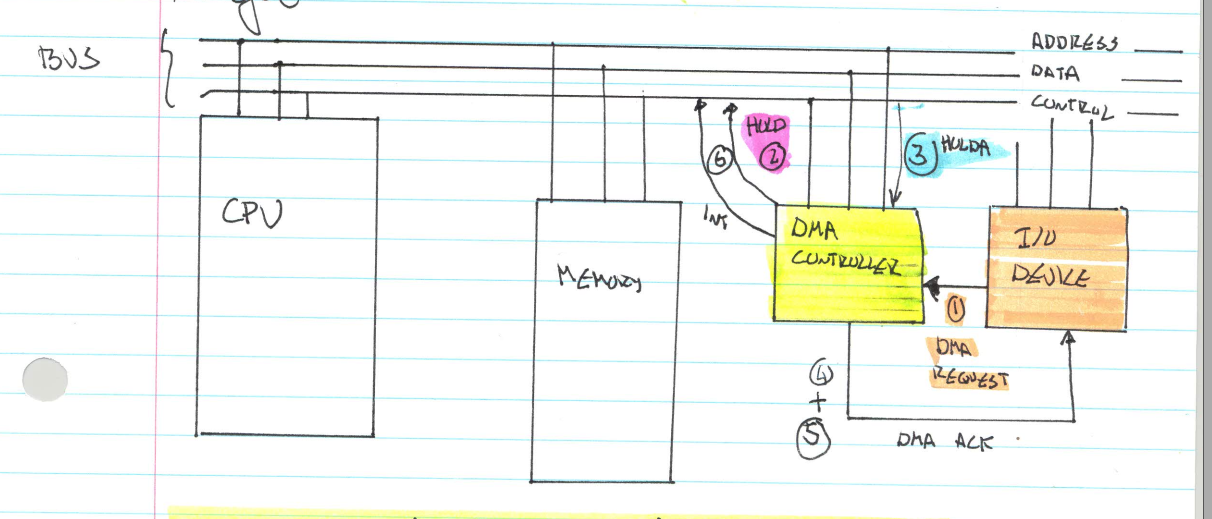
接下来的电路决定了哪个优先级最高，一旦决定了优先级，CPU按次序对DEVICE发出inta信号，把isr 给hardware



DIRECT MEMORY ACCESS DMA

DMA是一种方法来传递IO设备和memory中间的data 不经过CPU。这个技术要经过一个

Dma controller



IO设备发出一个DMA申请信号给DMA controller

第二步 DMA 激活 CPU HOLD PIN，这是一个对CPU的申请让他放弃对BUS的控制

第三步，CPU给DMA 发一个 HOLD ACKNOWLEDGEMENT 告诉他可以使用BUS了

第四步 DMA CONTROLLER把他内部存储器比如说地址REGISTER发到BUS上，并给IO DEVICE 发一个 DMA ACKNOWLEDGEMENT

第五步，DMA CONTYROLLER 完成对数据的传输

第六步，给CPU发信号告诉 操作已完成，CPU重新控制BUS

DMA TRANSFER的分类

BLOCK TRANSFER 。一整个BLOCK的 TRANSFER，在这个BLOCK没传递完之前，CPU无法控制BUS，但是CPU可以进行内部操作，不需要BUS

CTCLE STEALING: DMA CONTROLLER从CPU偷了CLOCK CYCLE