Summary for COD lab 6

Tobiichi Origami

September 7, 2020

Chapter 1

系统总线[1]

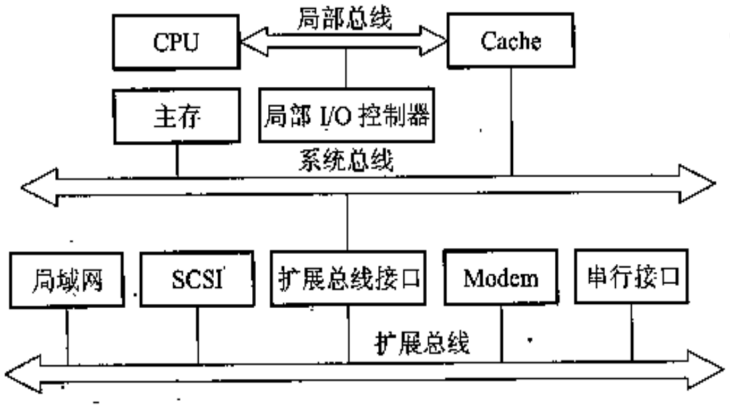
1.1 总线标准[2]

所谓总线标准，可视为系统与各模块、模块与模块之间一个互联的标准界面。这个界面对它两端的模块都是透明的……按总线标准设计的接口都可以视为通用接口

重点关注：PCIe 总线、USB 总线、type-C 线和 lighting 线?

1.2 总线结构[3]

参考这张图（其中的扩展总线接口用 DMA）



所以需要设计一个 cache（模块调用直接连到 MEM 段）

1.3 总线控制[4]

包括判优控制（或称仲裁逻辑）和通信控制

[1]唐书第三章
[2]Ch 3.3.3
[3]Ch 3.4
[4]Ch 3.5

1.3.1 总线判优控制

分为集中式和分布式两种：集中式又包括链式查询 (会出现饿死的情况)、计数器定时查询 (可以通过计数器不清零来实现没有优先级)、独立请求 (硬件复杂)

1.3.2 总线通信控制

一次总线操作分为 4 个周期：申请分配、寻址、传数、结束 (对于仅有一个**主模块**的简单系统，无需申请、分配和撤除)。通信又分为同步和异步，同步通信有着固定的传输周期任务规定[5]，并 (可能采用 SYN 标识开始)；而异步通信采用握手的方式，分为不/半/全互锁三种 (即一/二/三次握手)，对于传数的数据有打包处理 (一帧 = 1 起始位 (低电平) + 5～8 个数据位 +1 个奇偶校验位 + 1/1.5/2 个终止位 (高电平))，任意两帧之间任意长度高电平。类似的还有半同步通信[6]和分离式通信。

异步通信的应答　异步通信的应答方式可分为不互锁、半互锁和全互锁三种类型：

不互锁　主模块（对总线有控制权的模块）发出请求信号之后，不必等待接到从模块的回答信号，而是经过一段时间自动撤销；从模块接到请求信号之后，*在条件允许时发出回答信号*，且经过一段时间自动撤销回答信号。例如，CPU 向主存写入信息

半互锁　主模块发出请求信号，必须在*等到从模块的回答信号*之后才能撤销请求信号；而从模块在接到请求信号之后发出应答信号，但是经过一段时间就直接撤销。例如，在多机系统中，某个 CPU 需访问共享存储器时，该 CPU 发出访存命令后，必须接收到存储器未被占用但回答信号，才可以进行访存

全互锁　双方都采用等待应答的方式。例如网络通信

[5]唐书 P60，电子书 P72
[6]唐书 P64，电子书 P76

Chapter 2

输入输出系统[1]

2.1 概述

主机和 I/O 设备之间联络的方式，分为：

(1) 立即响应

(2) 异步工作采用应答信号联络，如前所述，用起始和终止信号来建立主从设备之间的联系

(3) 同步工作采用同步时标联络

异步工作如图所示：如同

并行传送的异步联络方式。

如图 5.6 所示，当 CPU 将数据输出到 I/O 接口后，接口立即向 I/O 设备发出一个“Ready”（准备就绪）信号，告诉 I/O 设备可以从接口内取数据。I/O 设备收到“Ready”信号后，通常便立即从接口中取出数据，接着便向接口回发一个“Strobe”信号，并让接口转告 CPU，接口中的数据已被取走，CPU 还可继续向此接口送数据。同理，倘若 I/O 设备需向 CPU 传送数据，则先由 I/O 设备向接口送数据，并向接口发“Strobe”信号，表明数据已送出。接口接到联络信号后便通知 CPU 可以取数，一旦数据被取走，接口便向 I/O 设备发“Ready”信号，通知 I/O 设备，数据已被取走，尚可继续送数据。这种一应一答的联络方式称为异步联络。

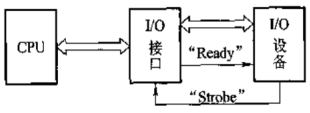


图 5.6 异步并行“应答”联络方式

2.1.1 I/O 设备与主机信息传送的控制方式

这个超级重点

共有 5 种方式：程序查询方式、程序中断方式、直接存储器存取方式 (DMA)、I/O 通道方式、I/O 处理机方式。主要用前三种，下面对前三种逐一整理

(1) 程序查询方式

当现行程序需要启动某 I/O 设备工作时，将查询程序插入到正在运行到程序中。只要一启动 I/O 设备，CPU 就不断查询 I/O 设备的准备情况，从而终止了原程序的执行。这种方式相当于进程调度，context switch 踢出去原有进程进入 waiting 状态，然后执行其所需的 I/O 进程 (貌似略去了 ready 状态)。查询通过做 *I/O* 指令完成：对 *I/O* 接口的某些控制寄存器置“*0*”或“*1*”，用于

[1]唐书第五章

控制设备进行相关工作；测试设备的某些状态，如“忙”、“就绪”等，以便决定下一步操作；传送数据

(2) 程序中断方式

CPU 在启动 I/O 设备后，继续执行现有程序，只是 I/O 设备准备就绪向 CPU 发出中断请求后才予以响应。这个方式是中断处理，需要保护现场 (当然前面那个也需要)，详情见后面中断处理部分。程序中断方式不仅需要在硬件上增加相应的电路，而且在软件上面也需要编制相应的中断服务程序 (详见 5.3 和 5.5)

(3) DMA 方式

(5.6)

关于程序中断方式等操作等一点 Notes

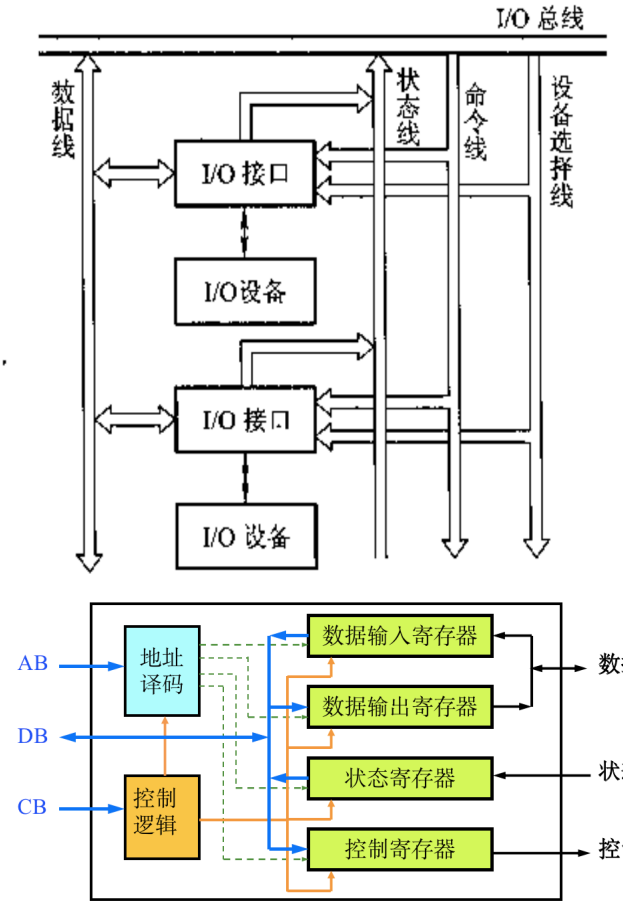
注意这张图里面第三行的“尚未”应当去掉。意即在一次中断方式的 *I/O* 操作中，首先如果待传送数据过多，则拆分到不同的存取周期来传送 *(*可以采用并行传输，二者并不影响*)*；其次在 *CPU* 需要用到 *I/O* 时，先启动并继续执行现有程序，等待设备的中断到达之后，开始执行 *I/O* 的取数据 *(*从接口的 *buffer* 经 *CPU* 传送至内存*)*；在这个传输过程中，可能会发生诸如异常处理中断之类的，需要按照中断处理机制嵌套递归；否则待这一次传输全部结束之后，才开始后面的 *CPU* 指令执行 *(*可以采取关闭中断的措施，如互斥锁、*MASK)*

由图中可见，CPU 向 I/O 设备发读指令后，仍在处理其他事情（如继续在算题），当 I/O 设备向 CPU 发出请求后，CPU 才从 I/O 接口读一个字经 CPU 送至主存（这是通过执行中断服务程序完成的）。如果 I/O 设备的一批数据（一个数据块的全部数据）尚未传送结束时，CPU 再次启动 I/O 设备，命令 I/O 设备再做准备，一旦又接收到 I/O 设备中断请求时，CPU 重复上述中断服务过程，这样周而复始，直至一批数据传送完毕。

2.2 I/O 接口

使用接口的理由 (接口的作用):

数据缓冲 (读取速度不一样)、串 - 并格式的转换 (CPU 并行)、电平转换、传送 (CPU 到 I/O 的) 控制命令、监视设备工作状态 (忙、就绪、错误、中断请求)。注意端口 (Port) 是接口 (Interface) 电路中的一些寄存器 (接口还包括控制逻辑电路)，包括数据端口、控制端口、状态端口。接口与总线、设备的连接如图所示:



现代计算机大多采用三态逻辑电路来构成总线。

接口的功能和组成：

选址功能 (片选，设备选择线上的设备码与本设备的设备码相符时，发出设备选中信号 SEL)、传送命令的功能 (设有命令寄存器 (存放 I/O 指令中的指令码，受 SEL 控制写使能) 和命令译码器)、传送数据的功能 (有数据暂存寄存器)、反映 I/O 设备工作状态的功能 ($^{eg.}$ 用完成触发器 D 和工作触发器 B 来标志设备所处的状态：D = B = 0 为暂停状态；D = 1, B = 0 为就绪状态；D = 0, B = 1 为正忙状态。还可以另外设置中断请求触发器 INTR (为 1 发出中断请求) 和屏蔽触发器 MASK (与 INTR 配合使用 (8.4)))

2.3 程序查询方式

2.3.1 程序查询流程

当 I/O 设备较多时，优先级查询 (轮询)。通常要执行三条指令：测试指令 (测试设备是否准备就绪)、传送指令 (处理 I/O 请求)、转移指令 (未就绪，转移到下一个设备询问)。程序流程如下图所示：

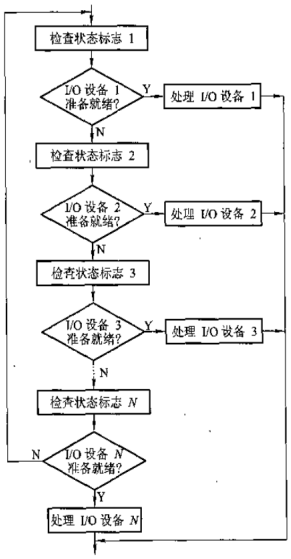


图 5.33 多个 I/O 设备的查询流程

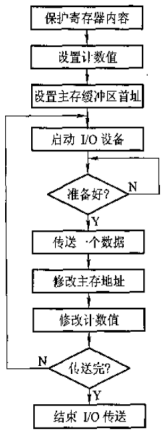
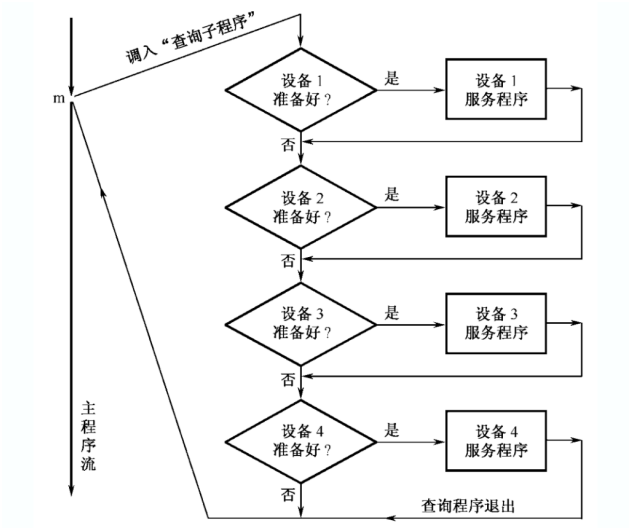


图 5.34 程序查询方式的程序流程



该方式必须将 I/O 查询程序插入到现有到程序中，故此方式包括以下几项：

① 由于需要占用 CPU 的寄存器，所以需要先保护起来

② 由于转送是一批数据，因此需要先设置 I/O 设备与主机交换的计数值

③ 设置欲传送数据在主存缓冲区的首地址

④ CPU 启动 I/O 设备

⑤ 将 I/O 设备中的设备状态标志取至 CPU 并测试设备是否就绪 (在设备接口电路里面输入缓冲满或者输出缓冲空)，没有就等着

⑥ CPU 执行 I/O 指令，读或者写一个数据，并将设备的状态标志**复位**

⑦ 修改主存地址

⑧ 修改计数值 (原码 –；负数的补码 ++)

⑨ 判断计数值，若不为 0，重新启动外设 (即跳转至 ⑥)

⑩ 结束 I/O 传送，继续执行现有程序

2.3.2 程序查询方式的接口电路[2]

2.4 程序中断方式

有关中断的详细内容在第八章

2.4.1 程序中断方式的接口电路

设备部件：

1. 中断请求触发器和中断屏蔽触发器

当且仅当设备准备就绪 (D = 1)，且该设备未被屏蔽 (MASK = 0)，CPU 的中断查询信号可以 (INTR = 1)

2. 排队器

就 I/O 中断而言，速度越高的设备优先级越高。硬件排队器可以在 CPU 里面设置一个计数器 (图 8.25)，也可以用链式排队器，如图所示：

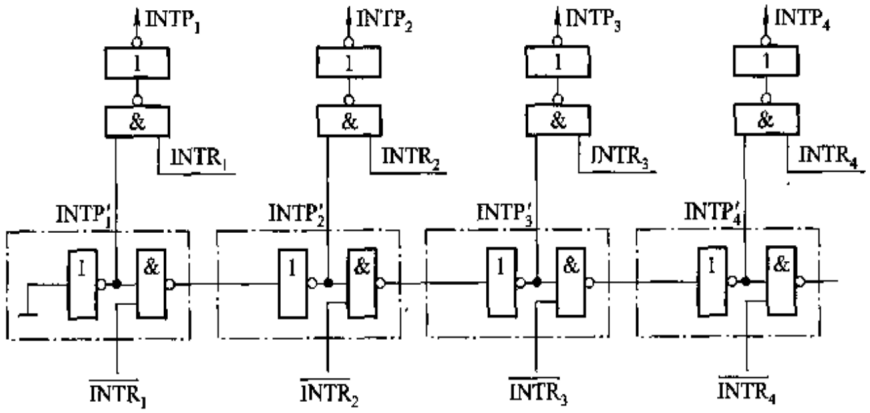


图 5.38 链式排队器

当各个中断源均无中断请求时，各个 $\overline{\mathbf{INTR_i}}$ 为高电平，其 $\mathbf{INTP_i'}$。当某个中断源提出中断请求时，就会迫使优先级比他低的中断源的 INTP 变为低电平，从而阻止其中断请求

3. 中断向量地址形成部件 (设备编码器)

中断向量形成部件的输入是来自排队器的输出 $\mathbf{INTP_1}$,...$\mathbf{INTP_n}$，而其输出则是一个中断向量 (二进制代码表示)，其实质是一个编码器。这里必须分清**向量地址**和**中断服务程序的入口地址 (都在主存里面)** 是两个不同的概念，他们之间的关系类似于一个哈希映射，如图所示：

[2]唐书 P192，电子书 204

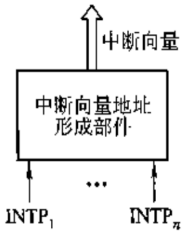


图 5.39 中断向量地址形成部件框图

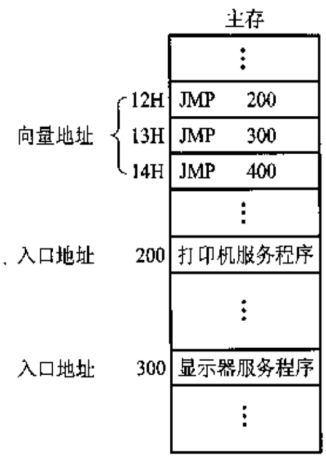
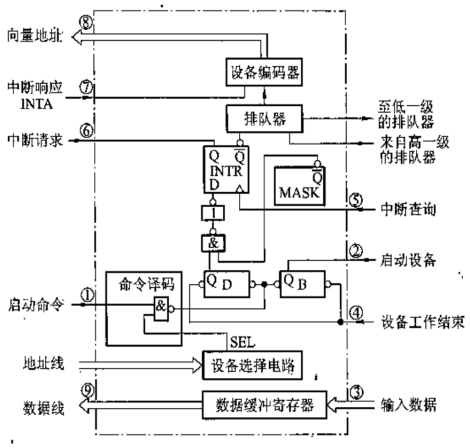


图 5.40 通过向量地址寻找入口地址

4. 程序中断方式接口电路的基本组成 (如图)



各问题的处理方式：(来自 PPT)

(a) 各中断源如何向 CPU 提出请求?(INTR, INTA)

(b) 各中断源同时提出请求怎么办? (中断判优-程序查询，硬件排队-集中、分布)

(c) CPU 什么条件、什么时间、以什么方式响应中断? (指令周期结束，中断隐指令)

(d) 如何保护现场?(中断隐指令断点、ISR、堆栈)

(e) 如何寻找入口地址?(硬件向量，软件查询)

(f) 如何恢复现场，如何返回?(中断隐指令断点、ISR、堆栈)

(g) 处理中断的过程中又出现新的中断怎么办? (多重中断，可屏蔽，设置屏蔽字)

2.4.2 程序中断方式的 I/O 中断处理过程

1. CPU 相应中断的时间和条件

   CPU 响应 I/O 设备提出中断请求的条件是必须满足 CPU 中的允许中断触发器 (互斥锁) 为 1。

该触发器可用开中断指令置位，也可以用关中断指令或者硬件使其自动复位。由于 I/O 准备就绪的时间 (D = 1) 是随机的，而 CPU 是在统一的时刻 (EX 段结束前) 向接口发中断查询信号，以获取 I/O 的中断请求。故 CPU 响应中断的时间一定是在执行阶段的结束时刻

2. I/O 中断处理流程 (以输入设备为例)
   唐书 P198，电子书 P211

3. 中断服务程序的流程

   (a) 保护现场

   (b) 中断服务

   (c) 恢复现场

   (d) 中断返回

2.5 DMA 方式

特别适用于高速 *I/O*

在 DMA 方式中，数据传送前后的准备和处理工作，由 CPU 上的管理程序负责，DMA 控制器仅负责数据传送工作

2.5.1 DMA 与主存交换数据时采用如下三种方法：

1. 停止 CPU 访问主存 (在外设要求传送的时候 CPU 就放弃，然后一直 stall)

2. 周期挪用/周期窃取 (读书人的事)；当 DMA 要求访问主存时，分为如下三种情况

   (a) available (CPU 不在也不打算访存)：直接访问内存

   (b) busy (CPU 正在访存)：等着

   (c) 访存冲突：CPU 在执行访存指令的过程中插入了 DMA 请求，并挪用了一、二个存取周期，使 CPU 延缓了一、二个存取周期再访问主存

3. DMA 与 CPU 交替访问 (Round-Robin)

访存冲突的原因其实是对同一个目标的写冲突

Chapter 3

中断系统[1]

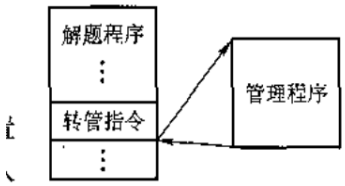
3.1 理论部分 (唐书)

3.1.1 概述

1. 引起中断的各种因素

(a) 人为设置的中断

一般称为自愿中断，执行策略如图所示：



自愿中断

(b) 程序性事故，可能发生的异常如图所示 (常见的)

0: Interrupt，中断；
1: TLB Modified，试图修改TLB中映射为只读的内存地址；
2: TLB Miss Load，试图读取一个没有在TLB中映射到物理地址的虚拟地址；
3: TLB Miss Store，试图向一个没有在TLB中映射到物理地址的虚拟地址存入数据；
4: Address Error Load，试图从一个非对齐的地址读取信息；
5: Address Error Store，试图向一个非对齐的地址写入信息；
6: Instruction Bus Error，一般是指令Cache出错；
7: Data Bus Error，一般是数据Cache出错；
8: Syscall，由syscall指令产生。操作系统下，通用的由用户态进入内核态的方法。
9: Break Point，由break指令产生。最常见的bp指令，是由编译器产生的，在除法运算时插入一个break point指令，以达到在除0时抛出错误信息的目的。因此，如果在定位问题时发现了一个Break Point异常，且它的异常分代码为07，应当考虑是出现了除0的情形；
10: RI，保留指令。在CPU执行到一条没有定义的指令时，进入此异常；
11: Co-processor Unavailable，协处理器不可用。这个异常是由于试图对不存在的协处理器进行操作引起的。特别的，在没有浮点协处理器的处理器上执行这条命令，会导致这个异常。随之，操作系统会调用模拟浮点的lib库，来实现软件的浮点运算；
12: **Overflow**，算术溢出。只有带符号的运算会引起这个异常；
13: Trap，这个异常来源于trap指令。和syscall指令类似地，trap指令也会引起一个异常，但trap指令可以附带一些条件，这样可以用于调试程序用。
14: VCEI，指令高速缓存中的虚地址一致性错误。
15: Float Point Exception，浮点异常；
16: Co-processor 2 Exception，协处理器2的异常；
17~22，留作扩展；
23: Watch，内存断点异常。当设定了WatchLo/WatchHi两个寄存器时起作用。当load/store的虚拟地址和WatchLo/WatchHi中匹配时，会引发这样一个异常
24~30，留作扩展；

嵌入式系统实验室 EMBEDDED SYSTEM LABORATORY

异常

[1]唐书第八章第四节、COD Ch4.9

(c) 硬件故障

(d) I/O 设备请求中断

(e) 外部事件 (用户通过键盘来发出中断，如 ctrl+C/D/Z)

2. 中断系统需要解决的问题

(a) 各中断源如何向 CPU 提出中断请求

(b) 当多个中断源同时提出请求时，中断系统对其作出响应仲裁

(c) CPU 在什么条件、什么时候、以什么方式来响应中断

(d) CPU 在响应中断之后如何保护现场

(e) CPU 响应中断后，如何停止原程序的执行而转入中断服务程序的入口地址

(f) 中断处理结束后，CPU 如何恢复现场并返回到原程序的间断处

(g) 若在中断处理过程中又出现了新的中断请求，CPU 应该如何处理

3.1.2 中断请求标记和中断判优逻辑