doi:10.3969/j.issn.1008-6714.2015.07.042

最小模式下 8086 引脚的需求引导式教学法

王春兰
(甘肃政法学院 公安技术学院,兰州 730070)

摘　要:针对众多高校计算机类学生在"微机原理与接口技术"课程学习中,对需要重点掌握且熟练运用的最小模式下的 8086 的 40 个引脚这个知识点望而生畏的教学问题,分析了在设计和制作 8086 芯片时所面临的工艺局限和 8086 芯片的功能需求,提出了以需求为主线,按照 8086 芯片的功能实现来逐步引出 8086 的 40 个引脚,让学生在主动思考中掌握知识的重点难点。经过多年的实践证明,学生能快速掌握 8086 引脚并熟练应用。

关键词:8086;引脚;地址;数据;读写
中图分类号:TP301　　**文献标志码**:A　　**文章编号**:1008-6714(2015)07-0086-03

"微机原理与接口技术"是工科、尤其是计算机专业最重要的专业基础课之一。通过本课程的学习,学生可以获得微机软、硬件方面的基本理论和基本技能,从而使学生具备软、硬件开发的初步能力。然而,"微机原理与接口技术"课程教学内容多、涉及面广、信息量大、实践性强,是学生普遍认为难学的课程。很多知识点需要熟练记忆,并灵活运用。比如最小模式下 8086 芯片的 40 个引脚及其功能这个知识点非常重要,学生学习的时候觉得非常枯燥、难记。这就需要教师充分发挥主导作用,对知识的难点重点运用适当的教学方法。笔者在多年教学中向老教师学习请教,不断实践思考,总结了一些方法和经验,希望能起到抛砖引玉的作用。

学习 8086 引脚可以以自行设计 CPU 引脚的探讨方式来展开,即从需求出发来解决问题。我们可以让学生自行设计一个简单的 CPU 的基本功能,根据 CPU 的这些基本功能来引出实现这些功能所需要交换的信息,这些信息交换又通过相应的引脚来实现。所以在教学时教师首先引导学生设计 8086 的基本功能,然后再根据这些功能来引出 8086CPU 的引脚。

一、8086 引脚的基本功能

(一)读写存储器、读写 I/O 接口

因为 CPU 最基本的功能是执行程序,再具体一些,就是执行指令。根据冯·诺依曼思想,指令和数据都是存放在内存里,那么 CPU 要想完成指令的执行就需要从内存里取指令,从内存或 I/O 接口读操作数,再把运算结果存放到目标位置。并且,CPU 要读写的数据可能是 16 位,也可能是 8 位。

(二)中断处理

CPU 在执行指令时可能有其他紧急的、意外的事件发生,此时需要暂停当前的任务,转去处理这样的事件。这样的事件可能是紧急的、必须马上处理的,也可能是不太重要的、可以忽略。CPU 对不同类型的中断需要分别对待。

(三)总线管理

CPU 与主存和 I/O 接口的所有信息交换都要通过它们之间的总线来进行,通常来说由中央处理器 CPU 来控制这些总线。对于总线大家都知道,为了避免数据冲突,同时只能有一个控制器来掌权。可是,计算机系统里连接的其他控制器需要使用总线时,CPU 可能要出让总线使用权,并且在该控制器使用完毕再收回总线使用权,这就是说,CPU 需要具备总线管理的功能。

(四)与外部硬件同步

在一个计算机系统中,除了主机之外,往往还需要连接很多其他的硬件来扩充主机的功能,所以主机会和很多其他硬件进行连接、交换信息。但是外部硬件速度通常都比较慢,CPU 与外部硬件的信息交换要等到外部硬件就绪后才能进行,这个功能我们称作与外部硬件同步。

(五)其他的功能

CPU 需要具备的其他边缘功能,比如可以复位启动、工作模式的选择等。对 8086 来说,它可以工作在单机系统,也可以工作在多处理器系统,我们分别称作最小模式和最大模式。

以上 5 个功能模块,是每个 CPU 也是 8086 所具有的基本功能。当然,所有这些功能都要 CPU 和外界交换信息来完成,而引脚是芯片与其他部件交换信息的唯一途径。

下面就开始根据 8086 的基本功能来引导学生设计 8086 的引脚。在这个引导过程中,虽然看上去是让学生自主设计的一个过程,但是在同一个功能有几种方法可以实现时,就需要教师把学生往 8086 引脚结构上引导,就是说,引导过程要有一个主线。

收稿日期:2015-05-05
基金项目:甘肃政法学院教改项目(GZJG2014-B07)
作者简介:王春兰(1980—),女,重庆人,讲师,从事数字图像处理研究。

万方数据

二、8086 的引脚

首先，根据第一章已经学过的知识，学生已经了解到8086 受到当时制造工艺限制，只有 40 个引脚，而地址总线和数据总线分别为 20 根和 16 根，工作时钟为 5MHz。并且，学生也已经知道，在引脚根数受限的时候为了实现足够的功能，可以进行引脚复用。

接着引导学生进入正题：如何设计 8086CPU 的 40 个引脚才能实现上面提出的 5 个功能需求。

（一）基本引脚

首先，作为电子芯片，必须要有工作电压和接地引脚。所以 8086 要具备如下几个输入引脚：Vcc，GND。为了方便，8086 设计了 2 个 GND 引脚。另外，8086 还需要有工作时钟，用引脚 CLK 表示。这一下就有了 4 个引脚，还余下 36 个引脚。下面需要用这 36 个引脚来实现前面提出的 5 个功能。

（二）读写操作相关的引脚

1. 地址数据复用引脚 AD15 ~ AD0

要想完成对存储器、I/O 接口的读写操作，则需要有地址线、数据线和控制线。那么前面已经知道地址线是 20 根，数据线是 16 根，这两总线一加就是 36 根了，那前面剩下的 36 根就全部用完了，其他的功能就没法实现了，所以我们就应该想到要进行总线复用。地址信号是 A19 ~ A0，数据信号是 D15 ~ D0，那就把低 16 根地址引脚与 16 根数据引脚复用，即 AD15 ~ AD0（余下的四个地址引脚 A19 ~ A16 其实也与状态信号 S3 ~ S6 进行了复用，暂时不用提及）。这样地址线和数据线总共用掉了 20 根引脚，即 AD15 ~ AD0 和 A19 ~ A16。

2. 地址数据分离所需要的控制引脚

地址线和数据线的复用节省了很多引脚，新问题却出现了。在 CPU 的读写总线周期的 4 个 T 状态中，T1 时由 CPU 给出的地址信号在后续的 T2 ~ T4 期间要一直保持地址信号稳定，即在 T2 ~ T4 内存或 I/O 接口同时需要地址信号和数据信号，而复用的 AD 引脚上同时只能传输一种信号。所以下面需要解决的问题就是如何分离被复用的信号，这是需要重点详细讲解的地方。

引脚复用使得在同一根引脚上要传输两个信号，可是怎么分辨这个复用引脚上到底传输的是哪个信号呢？比如 AD15 ~ AD0，在某个时刻，在这复用引脚上到底传输的是地址信号还是数据信号呢？这个问题只有系统的主宰者 CPU 才知道答案，因为系统是在 CPU 控制下运作的。地址信号是由 CPU 送到地址总线上的，而数据信号也是 CPU 要求传输时才会出现在数据总线上的。在对主存和 I/O 接口的读写周期内，主存和 I/O 接口需要的是持续、稳定的地址信号和数据信号。我们要在 AD 复用线接到存储器和 I/O 接口之前，先把它们分离并保存起来才能给主存和 I/O 接口提供持续、稳定的地址信号和数据信号，这就需要增加两类辅助芯片了，它们分别是地址锁存器和数据收发器，分别用来传输地址信号和数据信号。CPU 把地址数据 AD 复用线同时接到这两个芯片时，从地址锁存器输出的就是地址信号，从数据收发器传输的就是数据信号。但是这两个芯片并不具备智能，没法判断当前 AD 线上到底是地址还是数据。它们只能根据 CPU 的命令来完成自己的工作。

所以，当 CPU 往地址数据复用总线上发送地址信号时，就要发出一个通知信号给地址锁存器，这个信号就是地址锁存允许 ALE（Address Latch Enable），这个信号就是告知地址锁存器，现在 AD 线上传输的是稳定的地址信号，要求它把当前的地址信号锁存下来，送给存储器和 I/O 接口。这样，存储器和 I/O 接口就可以在读写周期内获取所需的稳定的地址信号了。

剩下的问题就是，是不是 AD 线上传输的信号除了地址信号就是数据信号呢，也就是说 ALE 无效时 AD 线上就是数据信号？答案是否定的。因为总线同时连接多个部件，为了避免信号串扰和总线冲突，引脚采用的是三态电路，也就是引脚除了传输有效的高低电平信号外还有一个高阻状态。所以当 AD 复用线上可以传输数据信号时，也要发出一个数据允许信号$\overline{DEN}$，告诉数据收发器现在可以工作了，也就是当前 AD 线上传输的是数据信号。不过，跟地址信号不同，数据信号是双向的，所以 CPU 还要发出一个 DT/$\overline{R}$ 信号给数据收发器，告诉它当前 CPU 是在发送数据还是在接收数据。

那么通过 ALE、$\overline{DEN}$、DT/$\overline{R}$ 这 3 个控制信号，就可以从 AD 复用线上把地址信号和数据信号分离开来，向主存和 I/O 接口提供数据读写操作时所需要的稳定的地址和数据信号，如图 1 所示。

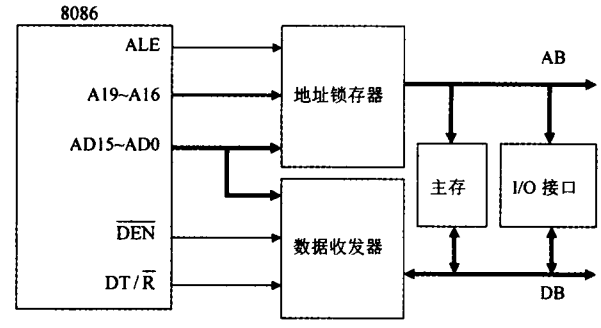

图 1　地址数据复用线分离示意图

读写数据需要的稳定持续的地址信号、数据信号有了，那么读写数据就可以实现了吗？答案是否定的。下面接着分析还需要有什么控制信号。

3. 读写数据所需要的其他控制引脚

首先，因为地址线和数据线同时连接到主存和 I/O 接口，而在读写数据时，同时只能读写一个对象，所以 CPU 读写数据时，就要告诉主存和 I/O 接口，当前读写的是哪一个对象，即需要一个控制信号来区分当前读写的对象是主存还是 I/O 接口，这种二选一的控制信号通常用一个引脚表示，即 M/$\overline{IO}$，由 CPU 送出，当它为高电平则表示读写主存，为低电平则表示读写 I/O 接口。

接下来，读写数据时有时是 16 位，有时只需要 8 位，而数据线有 16 位，所以 CPU 要用一个控制信号来通知主存和外设当前数据线的高八位是否传输有效数据，这个信号就是$\overline{BHE}$。

然后，对主存和 I/O 接口的操作可能是读，也可能是

万方数据

写，所以 CPU 还要用控制信号来通知主存或者 I/O 接口当前的操作类型，分别用$\overline{RD}$、$\overline{WR}$来表示。

还有，在读写外设时，因为一些外设太慢，为了不过分降低 CPU 的效率，我们在读写外设时，让外设在就绪时通知 CPU 去读写数据，即 READY 引脚，这个引脚是从外设传给 CPU 的。

小结一下，为了在引脚有限的情况下，完成 CPU 对主存和 I/O 接口的读写操作，需要有引脚 AD15 ~ AD0，A19 ~ A16，还有 8 个控制引脚 ALE，$\overline{DEN}$，DT/$\overline{R}$，M/$\overline{IO}$，$\overline{BHE}$，$\overline{RD}$，$\overline{WR}$，READY，总共是 28 个引脚。在图 1 中，把 8086 的这些引脚连接到内存和 I/O 接口，加上时钟发生器，即可得到 8086 的最小模式配置图。

（三）中断处理相关的引脚

为了实现中断处理，需要有中断请求信号和中断响应信号。中断请求分为必须马上处理的（不可屏蔽的），和可以处理也可以不理会的（可以屏蔽的）。为了便于 CPU 知道当前请求的中断的类型，就用两个引脚来引入不同类型的中断请求，即 NMI 和 INTR。NMI 表示外界部件向 CPU 发出的必须马上处理的、不可屏蔽的中断请求，INTR 表示其他部件向 CPU 发出的可以屏蔽的中断请求。那么对于可以屏蔽的中断请求，当前到底要不要处理呢？CPU 要有一个应答信号反馈给发出中断请求的部件，我们可以用$\overline{INTA}$来表示。通过这 3 个信号可以实现 CPU 的中断管理功能。

（四）总线管理

在正常情况下总线归 CPU 管理和使用，当系统里有其他的控制器需要使用总线时，就要向 CPU 申请总线的使用权，如果 CPU 允许，可以给该控制器一个反馈信号，表示把总线使用权借给该控制器使用，同时将三条总线置成高阻态。这一对信号就是 HOLD 和 HLDA。那么，在该控制器使用完毕的时候，是否需要再通知 CPU 收回总线使用权呢？没有必要为此多浪费一个引脚，我们让总线使用请求信号 HOLD 一直保持到控制器总线使用完毕，这样当 HOLD 变为无效时就表示该控制器不再使用总线了，CPU 则收回总线使用权，即让 HLDA 变成无效。

（五）与外部硬件同步

有时候 CPU 需要与外部硬件交换信息，但是外部硬件的速度往往比 CPU 要慢很多，CPU 需要等待。在等待过程中，CPU 每过一个时间间隔就去测试一下外部硬件是否准备好，如果还没有准备好，则再过一个时间间隔再次测试，如果已经准备好，则不再等待。这个过程我们叫做 CPU 与外部硬件的同步。供 CPU 测试的就是$\overline{TEST}$引脚，该引脚为外部硬件的就绪信号。一旦外部硬件就绪则通过该引脚输入低电平让 CPU 可以检测到。

（六）其他功能

8086 具有复位重启功能，用户的复位要求需要用一个引脚送入 CPU，这就是 RESET 引脚。另外，8086 可以工作于单机系统，也可以多个处理器协同工作，即最小模式和最大模式。8086 工作于单机系统时它负责提供系统所有的控制信号，而在多处理器系统中，则有专门的总线控制器来产生系统的控制信号。所有我们也要通过一个引脚告知 8086 它当前所处的工作模式，即引脚 MN/$\overline{MX}$。

到此为止，我们根据 CPU 的功能分析得出，CPU 具有 Vcc，GND（2 个），CLK，AD15 ~ AD0，A19 ~ A16，ALE，$\overline{DEN}$，DT/$\overline{R}$，M/$\overline{IO}$，$\overline{BHE}$，$\overline{RD}$，$\overline{WR}$，READY，NMI，INTR，$\overline{INTA}$，HOLD 和 HLDA，$\overline{TEST}$，RESET，MN/$\overline{MX}$共 40 个引脚。8086 的这 40 个引脚图如图 2 所示[1]。图中带“#”的引脚表示低电平有效。

实际上，我们上面的设计过程存在一个前提，那就是系统所需要的所有控制信号都是由 8086CPU 来产生的，也就是说 8086 工作于最小模式。而一旦 8086 工作于最大模式，则控制信号不再全部由它产生，所以前面分析得到的一些控制引脚也就发生了变化。这些变化体现在图 2 括号内。从图中可以看出，最大模式与最小模式下，8086 的引脚变化在于第 24 ~ 31 号共 8 个引脚，括号内是最大模式下的引脚含义。当然，对于最大模式中这些引脚的含义也可以采用相应的教学方法进行讲解，本文不再详述。

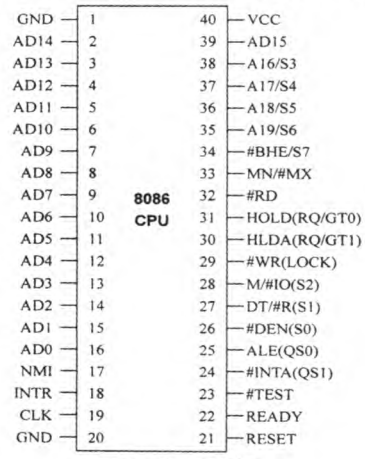

图 2　8086 引脚图

三、结束语

根据多年的教学经验，用需求引导法把枯燥繁多却又很重要的 8086 最小模式的 40 根引脚逐一讲解清楚。在教学实践中，用这种方法收到了良好的效果。通过两三个课时的课堂教学，学生可以基本掌握 8086 的这 40 个引脚的名称、功能和特点，并且理解了引脚复用的优点，同时掌握了 8086 的最小工作模式，为本课程后续的学习打下坚实的基础。

参考文献：

[1]周荷琴，吴秀清. 微型计算机原理与接口技术[M]. 合肥：中国科学技术大学出版社，2008.

[2]牟琦. 微机原理与接口技术[M]. 北京：清华大学出版社，2013.

[3]顾辉. 微机原理与接口技术——基于 8086 和 Proteus 仿真[M]. 北京：电子工业出版社，2011.

[4]钱晓捷. 16/32 位微机原理、汇编语言及接口技术教程[M]. 北京：机械工业出版社，2011.

[5]朱定华. 单片微机原理、汇编与 C51 及接口技术[M]. 北京：清华大学出版社，2013.

〔责任编辑：李海波〕

万方数据