有不少人问到：flash连接CPU时，根据不同的数据宽度，比如16位的NOR FLASH （A0－A19），处理器的地址线要（A1－A20）左移偏1位。为什么要偏1位？

(全文有点晦涩，建议收藏本文对照着1期MMU视频阅读)  
  
从软件和CPU的角度而言，一个地址对应一个字节，就是8位数据。这是肯定的，不要怀疑这点。  
  
对于具体器件而言，它的位宽是一定的，所谓位宽，指的是“读/写操作时，最小的数据单元”──别说最小单元是“位”，一般设备上没有单独的“位操作”，修改位时通过把整个字节、字或双字读出来、修改，再回写。  
  
CPU的地址线（A0－A20）对应的最小数据单元是字节，即8位；  
而位宽为16的NOR FLASH的地址线(A0－A19)对应的最小数据单元是16位。  
这两个怎么对应起来？  
  
如果说外设的位宽是16，难道我们写程序时会“特意”以16位进行操作吗？不用的，我们写程序时根本不用管外设位宽是8、16还是32。  
  
仔细想想，其实是可以想通的：**既然CPU、外设NOR FLASH的最小读/写单元已经固定，那么肯定就是CPU与NORFLASH之间有个中间层，**它来做处理：  
这个中间层被称为“Memory Controller”，CPU要进行读写操作时，“Memory Controller”根据NOR FLASH的位宽，每次总是读/写16位数据。  
以读操作为例：  
CPU想进行8位操作时，它选择其中的8位返回给CPU；  
CPU想进行16位操作时，它直接把这16位数据返回给CPU；  
CPU想进行32位操作时，它发起2次读/写，把结果组合成32位返回给CPU。  
  
现在的连线是：CPU的（A1－A20）接到 16位的NOR FLASH（A0－A19），即CPU的A0不接──这说明：**不管A0是0还是1，NOR FLASH接收到的地址是一样的**。

CPU发出地址0bxxxxxxxxx0、0bxxxxxxxxx1时，NOR FLASH看到的都是0bxxxxxxxxx，返回给“Memory Controller”的都是同一个16位数据。  
再由“Memory Controller”选择其中的低8位或高8位给CPU。  
  
“Memory Controller”会帮助我们做这些事情，举例为证：  
1. 软件要读取地址0上的8位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000000的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000000  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NORFLASH中的第1个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据的低8位返回给CPU，这就是一个8位数据。  
  
2. 软件要读取地址1上的8位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000001的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000000  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NOR FLASH中的第1个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据的高8位返回给CPU，这就是一个8位数据。  
  
3. 软件要读取地址2上的8位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000010的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000001  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NOR FLASH中的第2个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据的低8位返回给CPU，这就是一个8位数据。  
  
4. 软件要读取地址3上的8位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000011的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000001

② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NOR FLASH中的第2个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据的高8位返回给CPU，这就是一个8位数据。  
  
5. 软件要读取地址0和1上的16位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000000的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000000  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NOR FLASH中的第1个“最小数据单元”

③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据返回给CPU  
  
6. 软件要读取地址2和3上的16位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000010的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000001  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NORFLASH中的第2个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
④ “Memory Controller”把这个16位数据返回给CPU  
  
7. 软件要读取地址0、1、2、3上的32位数据时，硬件是这样进行的：  
① “Memory Controller”发出0b000000000000000000000的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000000  
② NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NORFLASH中的第1个“最小数据单元”   
③ “Memory Controller”读入这个16位数据  
     
④ “Memory Controller”发出0b000000000000000000010的地址信号，NOR FLASH的A0－A19线上的信号是：0b00000000000000000001

⑤ NOR FLASH在数据总线D0～D15上提供一个16位的数据，这是NORFLASH中的第2个“最小数据单元”   
⑥ “Memory Controller”读入这个16位数据  
⑦ “Memory Controller”把两个16位的数据组合成一个32位的数据，返回给CPU。  
     
从1～7可知：  
① **对于软件而言，它不知道底下发生了什么事，它只管结果**：  
读取地址0的8位数据，就得到了一个8位数据；读取地址1的8位数据，就得到另一个紧挨着的8位数据，读取地址0开始的16位数据，就得到了一个16位数据；读取地址2开始的16位数据，就得到另一个紧挨着的16位数据，读取地址0开始的32位数据，就得到了一个32位数据；读取地址4开始的32位数据，就得到另一个紧挨着的32位数据

② 对于NOR FLASH，它只按照A0－A19地址线，提供16位数据，才不管软件要的是8位、16位，还是32位呢。  
③“Memory Controller”完成了这些位宽之间的数据选择、合并。  
  
  
所以：  
外设位宽是8时，CPU的A0～AXX与外设的A0～AXX直接相连  
外设位宽是16时，CPU的A1～AXX与外设的A0～AYY直接相连，表示不管CPU的A0是0还是1，外设看到的都是同一个地址，对应16位的数据，“Memory Controller”对数据进行选择或组合，再提供给CPU。  
外设位宽是32时，CPU的A2～AXX与外设的A0～AZZ直接相连，表示不管CPU的A0A1是00，01，10还是11，外设看到的都是同一个地址，对应32位的数据，“Memory Controller”对数据进行选择或组合，再提供给CPU。