第四节 Intel 80486处理器

– 80486的主要改进:

① 增加了数据的猝发传送方式;

② 指令预取队列长度, 由16字节增加到32字节;

③ 片内集成了8K的Cache, 并支持片外Cache;

④ 片内集成了浮点协处理器FPU;

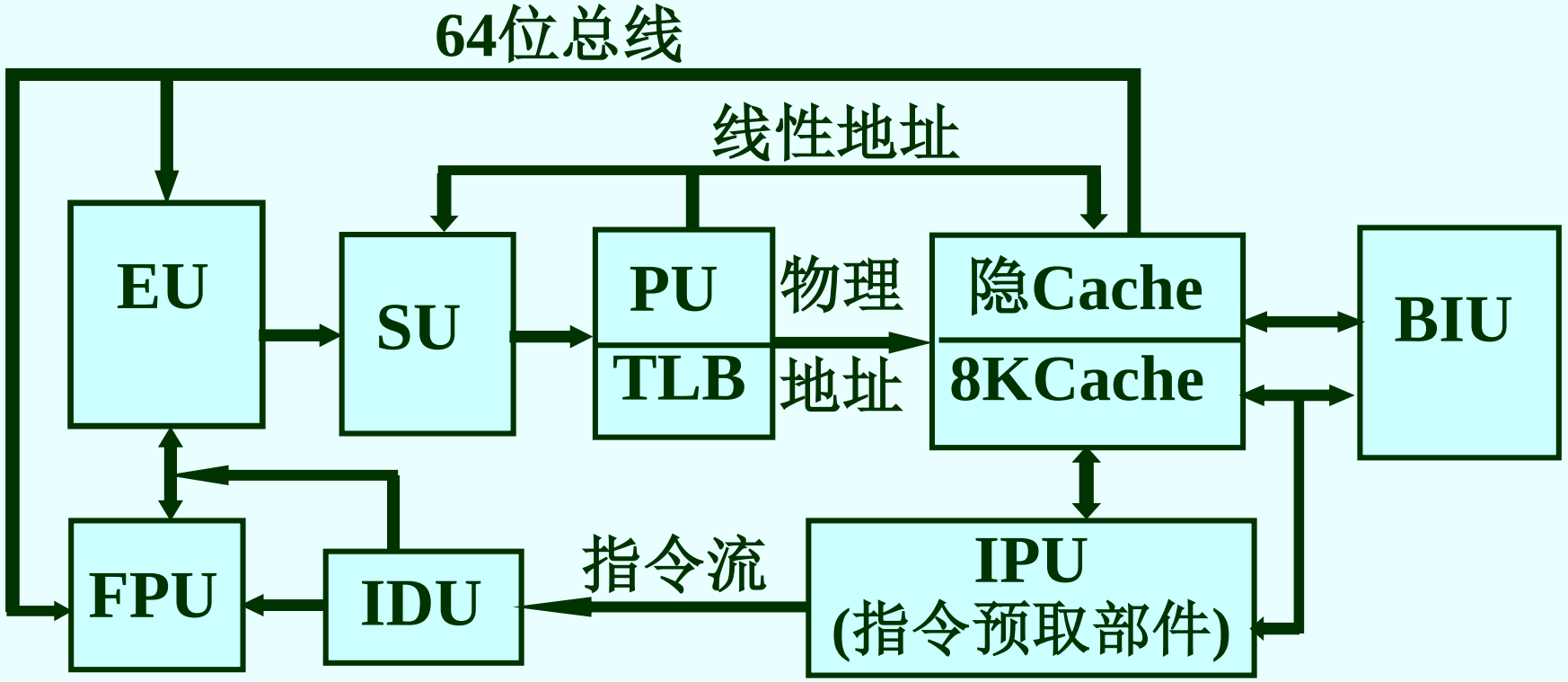
⑤ 支持数据位的奇偶校验;

⑥ 指令流水线方式

一、80486的内部结构

(一) 功能模块

在80386基础上, 增加: 8KCache, 协处理器 FPU



结构上, 除了FPU和Cache外, 与Intel80386基本相同

(二) 内部寄存器

通用寄存器/段寄存器/指令指针与80386相同。

1、FR标志位寄存器: 新增标志位

- AC对齐标志
  - 16位的字, 从偶地址开始存放(A_0为0);
  - 对32位的字, 起始地址能被4整除(A_1A_0 =00)

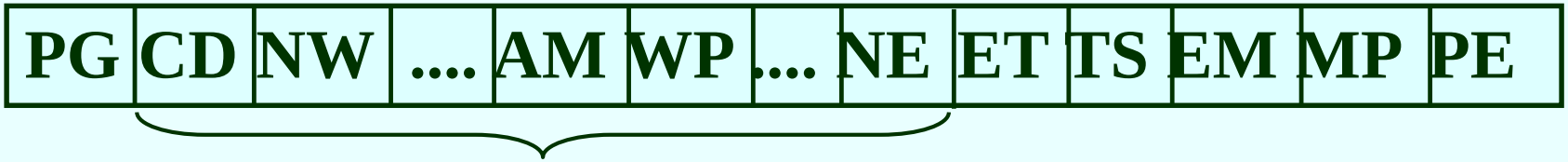
依此类推, 双字(64位)存放的起始地址$A_2A_1A_0$=000等。以上情况都称为对齐的。(这也是为什么CR_3、页目录和页表都只给出20位的基地址的原因。

AC = { 0 不作对齐检查; 1 进行对齐检查 } 仅在特权级3

进行对齐检查 → 未对齐时产生异常中断

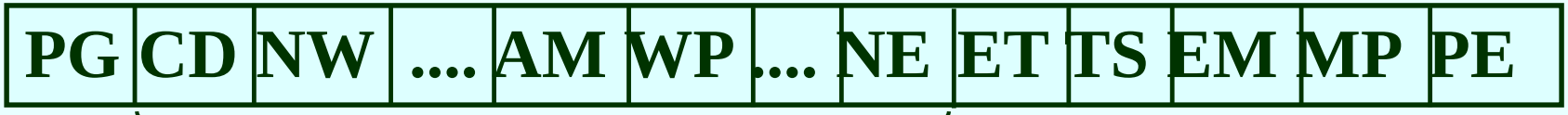
2、CR_0~ CR_3 控制寄存器

① CR_0



- CD
  - 1 读未命中时, 禁止填充Cache(即禁止从内存中读取的数据写入Cache)
  - 0 读未命中，允许填充Cache
- NW
  - 1 不允许直写 (写命中Cache, 不写入内存)
  - 0 允许直写Cache

注: 不可高速用存储器

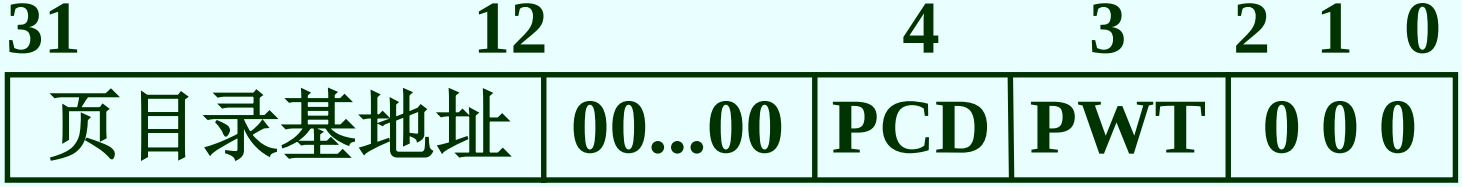


新增控制位

- **AM**
  - 1 允许AC对齐标志
  - 0 禁止AC对齐标志 (便于与80386兼容)
- **WP**
  - 1 任何特权级的任务, 对页面都只能读
  - 0 按描述子/页目录/页表的规定实施读写保护
- **NE**
  - 1 浮点部件出现异常时, 产生异常中断INT16
  - 0 且输入引脚$\overline{\text{IGNEE}}$有效, 忽略浮点部件出错

② CR_3

在80386基础上增加了两位:



- PCD
  - 1 禁止页目录项的内容进入Cache(禁止填充Cache)
  - 0 允许页目录项的内容进入Cache
- PWT
  - 1 片外Cache采用直写方式
  - 0 片外Cache采用回写方式

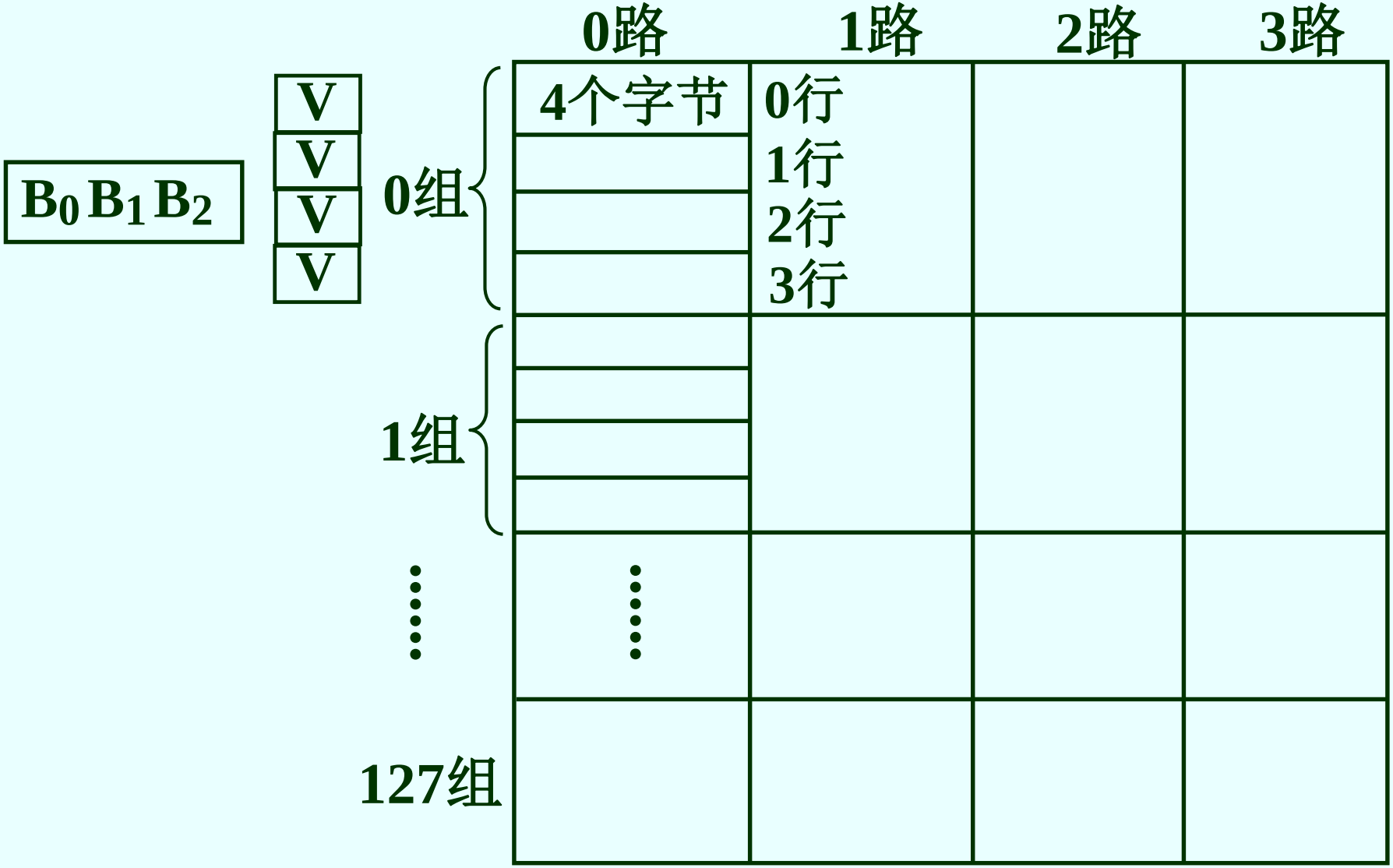
(注：80486片内的Cache只有直写方式)

二、80486片内Cache

80486片内集成8K的高速缓存。

(一) 结构

- 采用四组关联结构, 即分成四路, 每一路分成128组, 每组分成四行, 每行四个字节。
- 每一行设置了一个有效位V, 用于指示该行是否有效(V＝1时该行有效, V=0时该行无效)。
- 为了便于使用LRU算法, 对每四行, 设置了三个标志位B_0、B_1、B_2, 用来指示这四行最近被访问的情况。如下图所示:


0路
1路
2路
3路
V
V
V
V
$B_0 B_1 B_2$
0组
4个字节
0行
1行
2行
3行
1组
127组

– 每四行附加的标志位B_0、B_1、B_2指示这四行最近被访问的情况, 按以下方式进行设置:

最近被访问的行是
- 0行或1行, 则1→B_0
  - 0行 1→B_1
  - 1行 0→B_1
- 2行或3行, 则0→B_0
  - 2行 1→B_2
  - 3行 0→B_2

(二) 工作过程控制

1. 四种工作方式

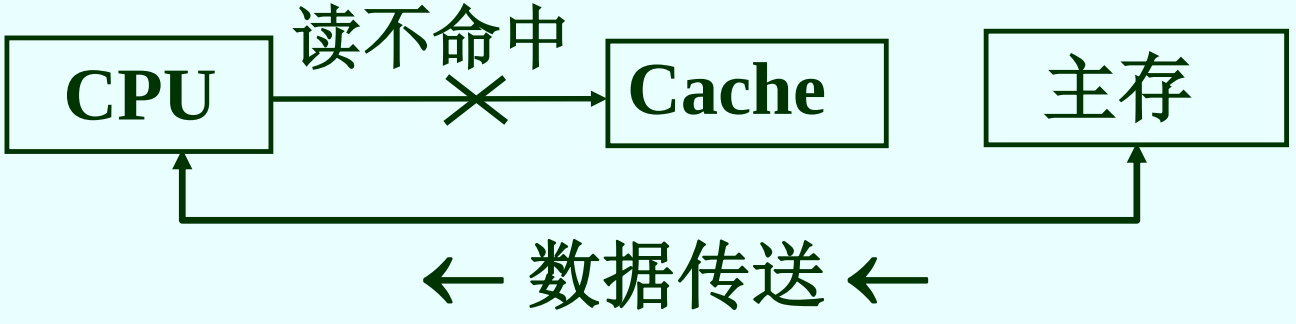
CR_0中CD和NW两个控制位, 其四种组合构成Cache的四种工作方式(注: 80486片内Cache,在“写不命中时, 只写内存, 不写Cache”):

CD	NW	功 能
0	0	正常工作方式(允许填充、直写和“使无效”功能)
0	1	无效(将该状态装入CR_0, 将产生异常中断)
1	0	禁止Cache填充, 但允许直写和“使无效”功能
1	1	禁止Cache填充, 禁止直写和“使无效”功能

说明:

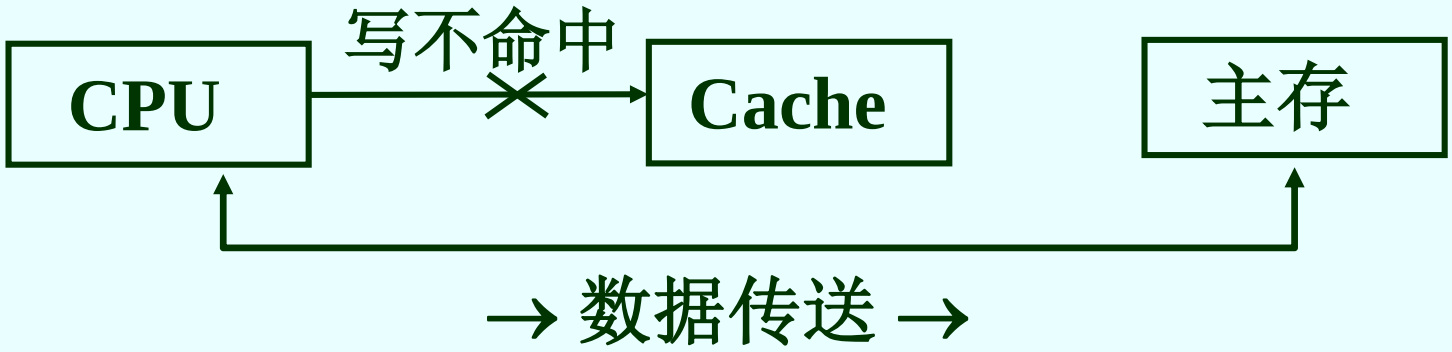
- CD和NW=10

  禁止Cache填充, 则在“读未命中时”只访问主存;



不允许从主存读取的数据和对应的地址写入Cache;

在“写不命中时”只写主存不写Cache。只允许直写(在“写命中”时, 既写入Cache, 又写入内存)



因此, CD和NW=10时, 可以用软件方式对某些主存单元的高速缓存性进行控制。比如,为防止Cache内容过时的方法之一，就是“不可高速用存储器”，禁止共享区的内存单元内容进入Cache。

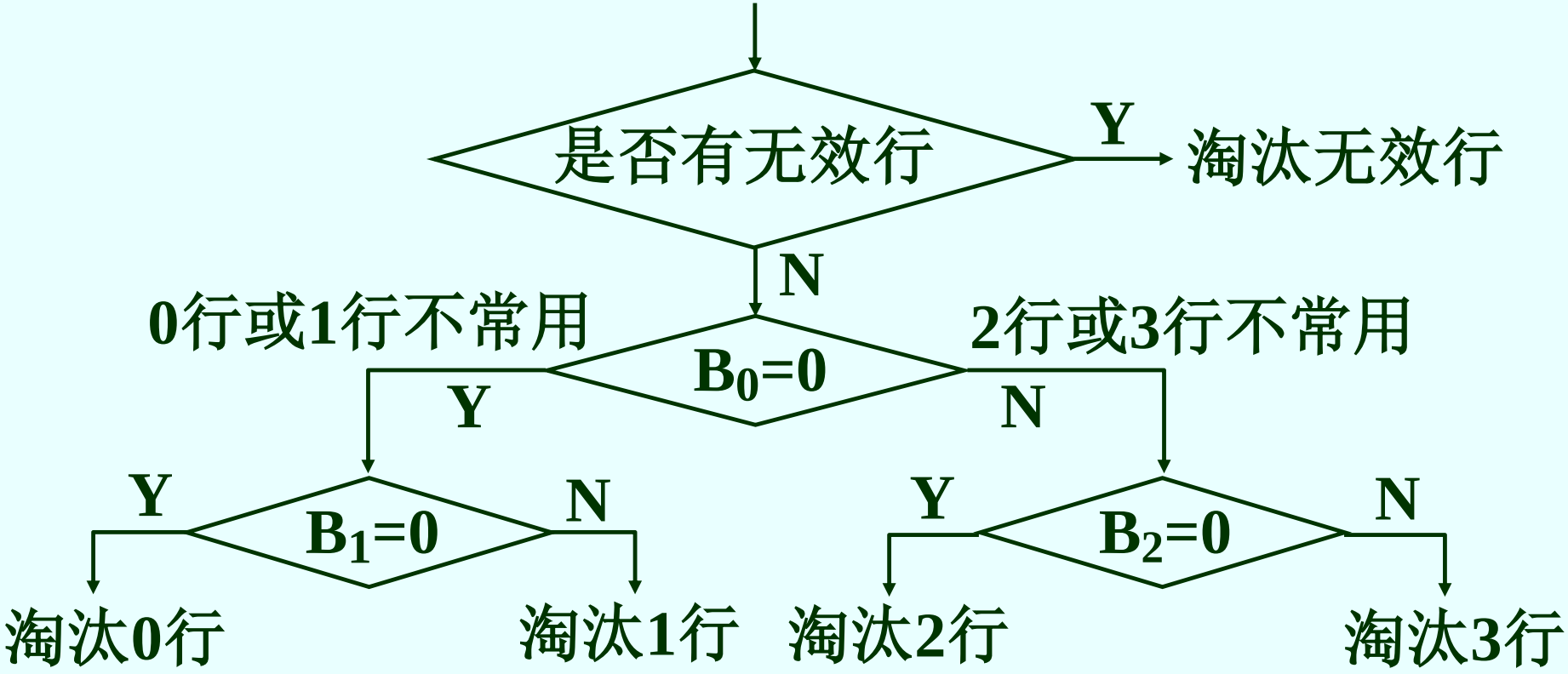
- **CD和NW=11时:**

  Cache的几种功能均被禁止，即“读不命中”时，只读主存而不写入Cache;“写不命中”时, 只写主存而不写入Cache, 且不允许直写; 也不能标识某些Cache内容无效, 意味着此时Cache与主存的联系已被断开。

  该功能主要用于程序测试。预先将某些测试所需要的数据装入Cache，然后将CD和NW设置为11，断开Cache与主存的联系。测试开始后，程序只访问预先装入Cache的数据，不会有启动总线周期访问内存, 也不会出现“不命中”。

2. Cache的更新(淘汰)

淘汰过程按如以下流程:



3. 高速缓存的清除

- 软件方式

  执行指令INVD或者WBINVD

  WBINVD先将Cache内容写回主存, 再清除。

  注: 80486片内的Cache采用的是直写方式, 这两条指令对片内Cache的执行结果相同; 但片外Cache可能采用回写, 所以设置了WBINVD指令。

- 硬件方式

  引脚 $\overline{\text{FLUSH}}$, 当该引脚有效时, 片内 Cache内容将被清除掉。

4. 页面的可高速缓存性

- 什么是页面的可高速缓存性?

  针对一个页面, 禁止或允许其内容进入Cache。

  (即控制页面的内容是否可以进入Cache,与“不可高速用存储器”的概念相同)。

- 如何控制页面的可高速缓存性?

① CR_0中的CD位

CD { 1 读未命中, 禁止填充Cache
   { 0 读未命中, 允许填充Cache

② CR_3控制寄存器中

PCD { 1 禁止页目录项的内容进入Cache
    { 0 允许页目录项的内容进入Cache

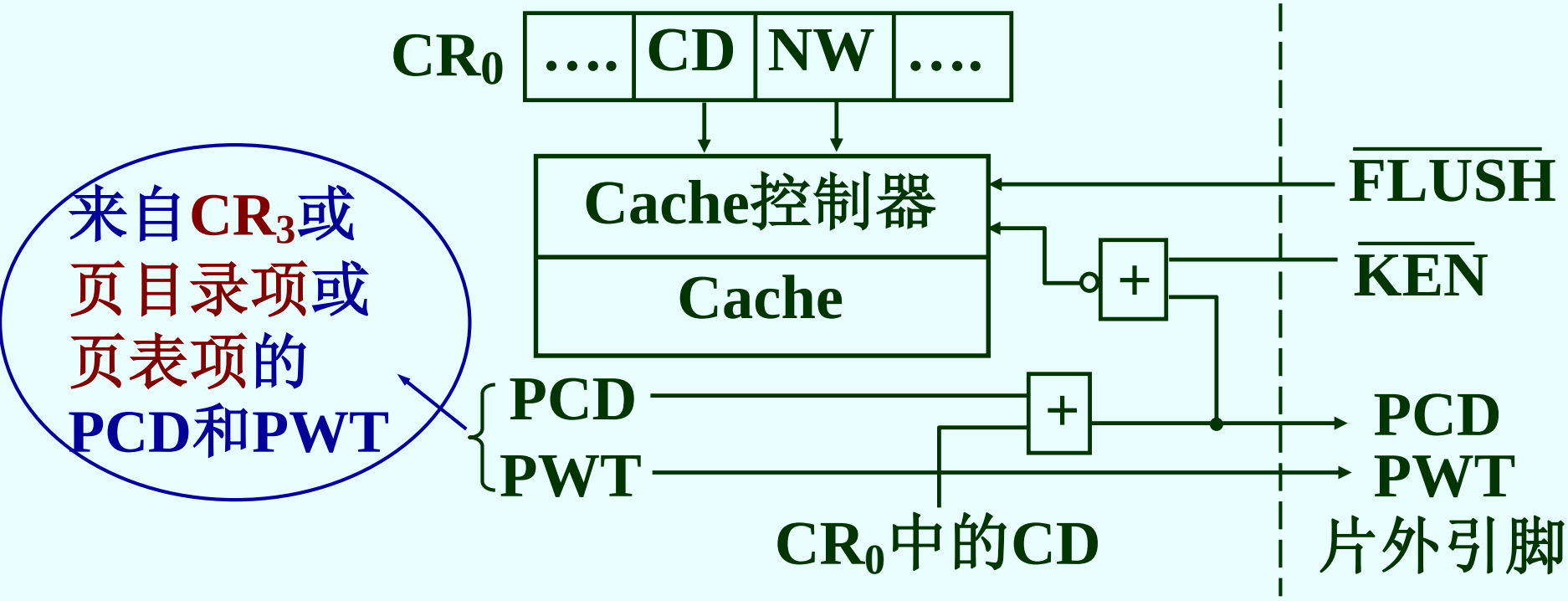
③ 80486的分页机制的页目录项和页表项中的PCD (D_4, 386处理器未使用该位):

PCD (页目录项)
- 1 禁止页表的内容进入Cache
- 0 允许页表的内容进入Cache

PCD (页表项)
- 1 禁止页面的内容进入Cache
- 0 允许页面的内容进入Cache

④ 80486的输出引脚 — 非独立功能PCD

上述信号及引脚之间的关系:



当“或非门”输出 { 1 当前页允许填充Cache
0 当前页不允许填充Cache

由此可得:

只有当CD=0、PCD=0、并且 $\overline{KEN}$= 0时，才能进行Cache填充, 三者中有一个为“1”，就禁止填充Cache。

三、80486的引脚功能

1. 数据奇偶校验引脚

DP_3~DP_0，每一位针对一个字节，即：

$\underbrace{D_{31}\sim D_{24}}_{DP_3}$ $\underbrace{D_{23}\sim D_{16}}_{DP_2}$ $\underbrace{D_{15}\sim D_{8}}_{DP_1}$ $\underbrace{D_{7}\sim D_{0}}_{DP_0}$

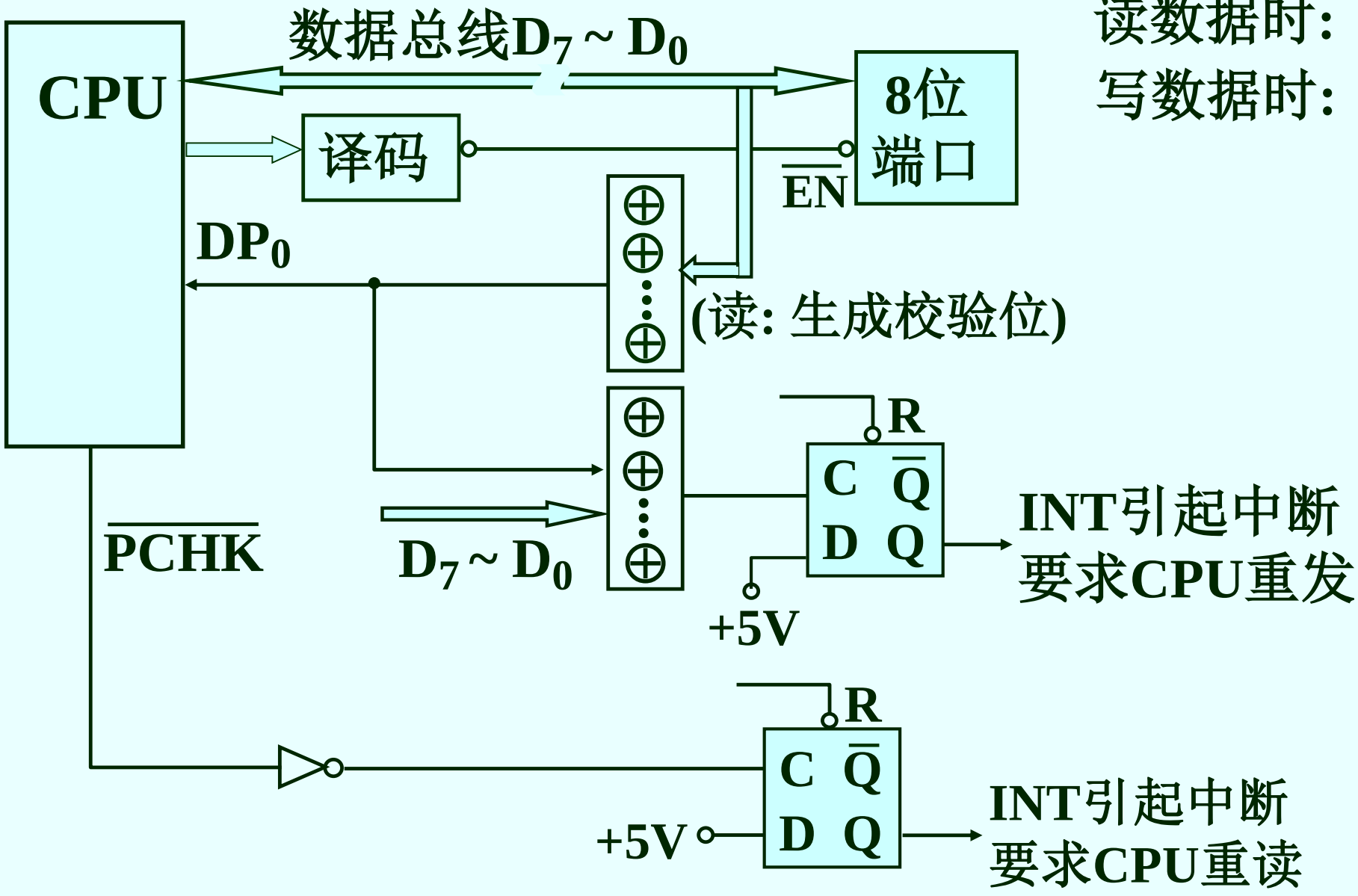
- 写数据时:
  CPU根据输出的数据，内部自动按偶校验方式生成DP_3～DP_0，并连同数据位一起输出。
- 读数据时:
  CPU将数据位连同校验位一起读回来。这要求被CPU读取的设备生成校验位。

2. 奇偶校验检查位(CPU输出) $\overline{PCHK}$

   读周期后有效, 指示读取的数据是否奇偶出错

   奇偶校验位及校验检查位 — 简化接口设计

   例: 处理器与一8位外设通信, 进行出错校验, 设计利用以上信号进行校验的接口电路原理图


读数据时:
写数据时:
CPU
数据总线D7 ~ D0
8位
端口
译码
EN
DP0
(读: 生成校验位)
R
C Q
D Q
INT引起中断
要求CPU重发
PCHK
D7 ~ D0
+5V
R
C Q
D Q
+5V
INT引起中断
要求CPU重读

3. 总线宽度控制信号 $\overline{BS16}$ $\overline{BS8}$

(1) $\overline{BS16}$ 请求进行16位数据传送

该信号与80386的$\overline{BS16}$的异同:

相同处:

当 $\overline{BS16}$ 有效, 表示外设请求CPU进行16位的数据传送。如果CPU执行的32位的I/O指令, 自动将其转换成两个16位的数据传送周期;

不同处:

对80386, 32位的数据只在总线低16位D_{15}~D_0上传送

对80486, 32位的数据会分别在总线的高16位和低16位上传送。

对80386，32位的数据只在总线低16位D_{15}~D_0上传送

读操作时:

CPU的两个16位读周期都只采样数据总线低16位;

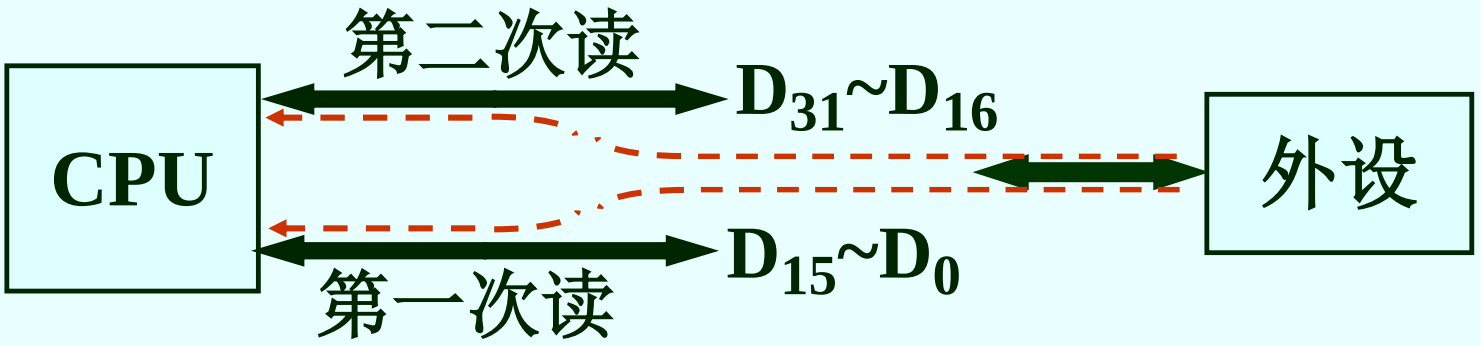
写操作时:

第一个周期，在总线的D_{15}~D_0上发送低半部;第二个周期，CPU将高半部自动复制到数据总线低16位D_{15}~D_0上发送。

只需将外设的16位数据线连接到系统数据总线的低16位D_{15} ~ D_0

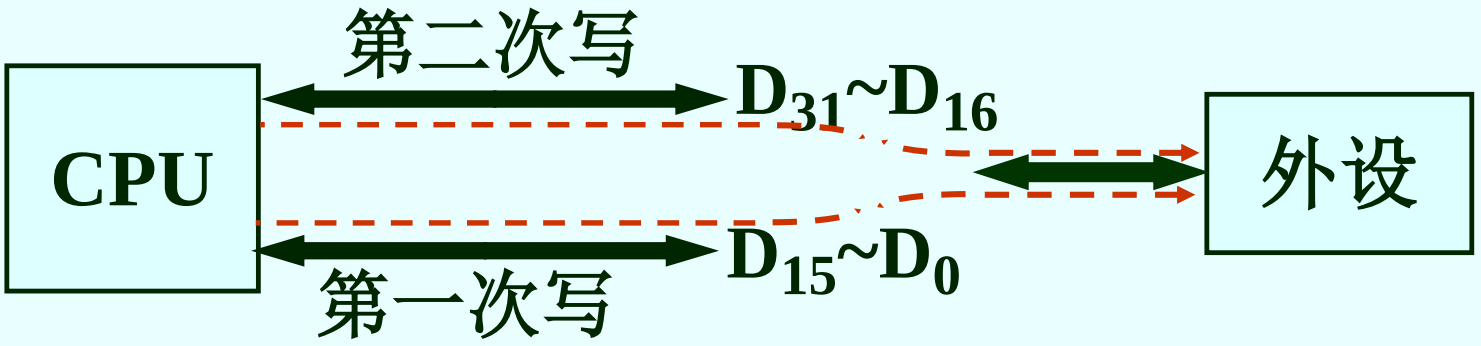
对80486, 如果CPU执行的32位的I/O指令, 第一个周期在数据总线低16位传送低位字, 第二个周期在数据总线高16位传送高位字。

- 读周期:



外设的16位数据线连接到系统数据总线的高16位D_{31}~D_{16}还是低16位D_{15}~D_0?

- 写周期:

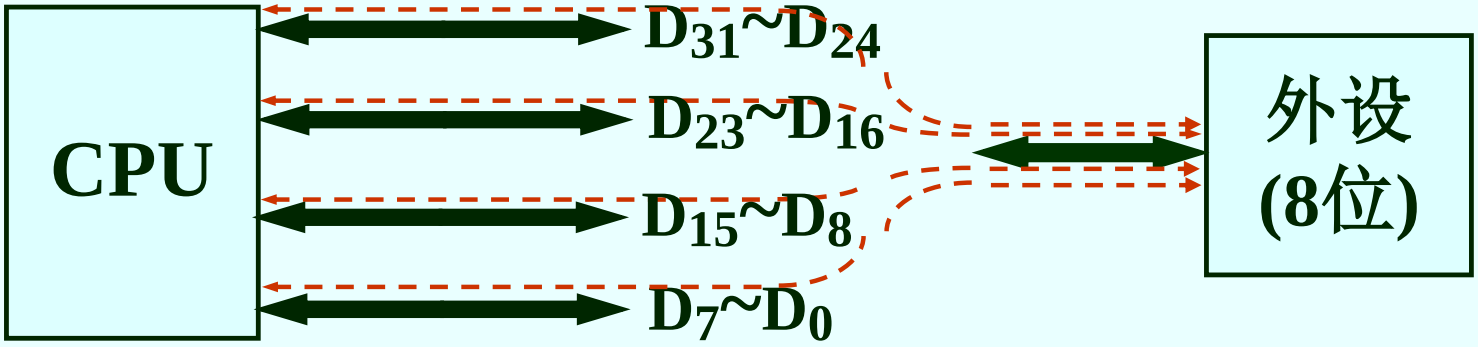


外设的16位数据线连接到系统数据总线的高16位D_{31}~D_{16}还是低16位D_{15}~D_0?

(2) $\overline{BS8}$

当 $\overline{BS8}$有效时,进行8位数据的传送,一次传送一个字节，针对8位外设。

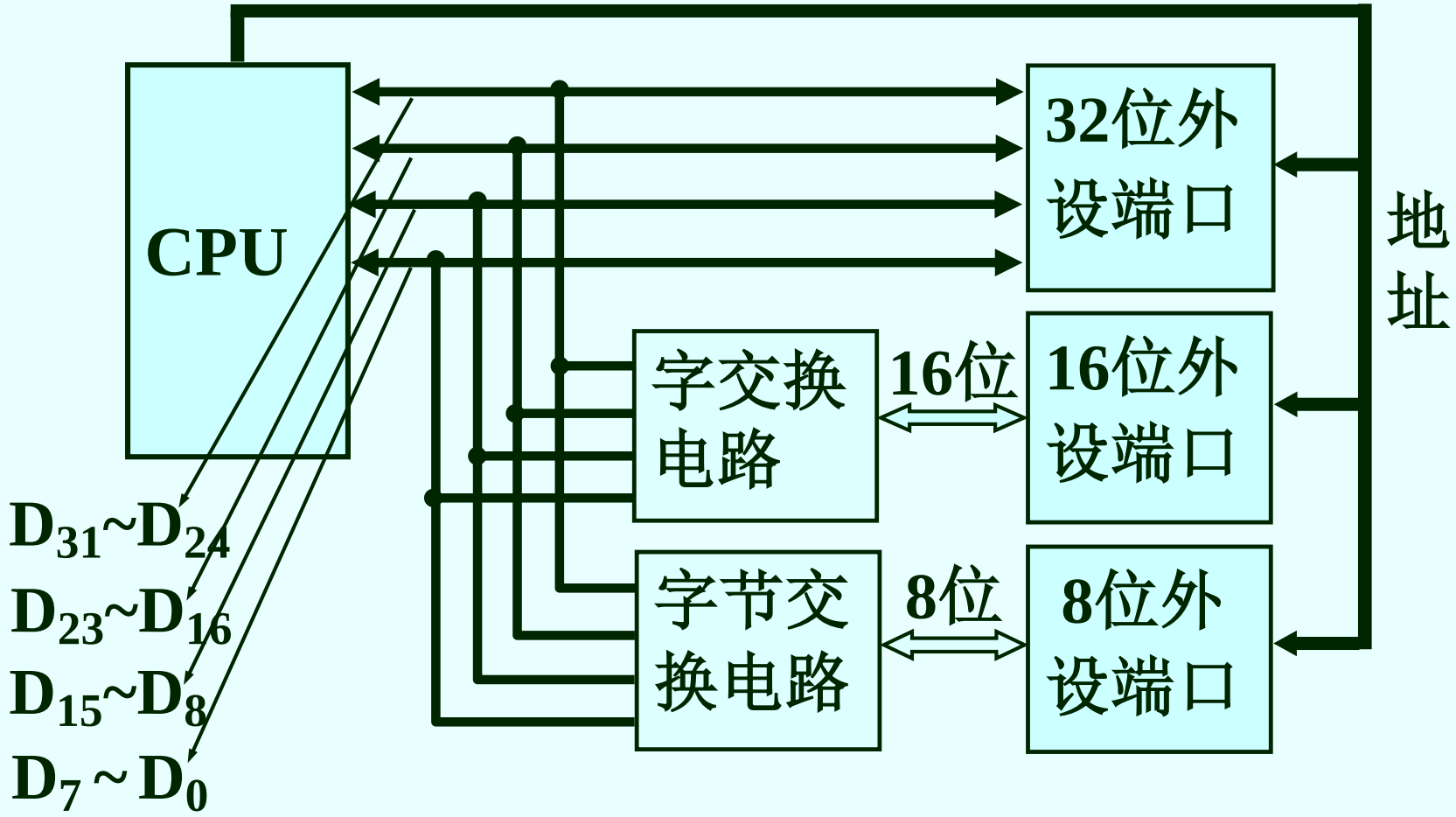
如果CPU执行的32位的I/O指令，则自动转换成4个总线周期。四个周期分别传送四个字节,依次在数据总线的四个字节上传送。



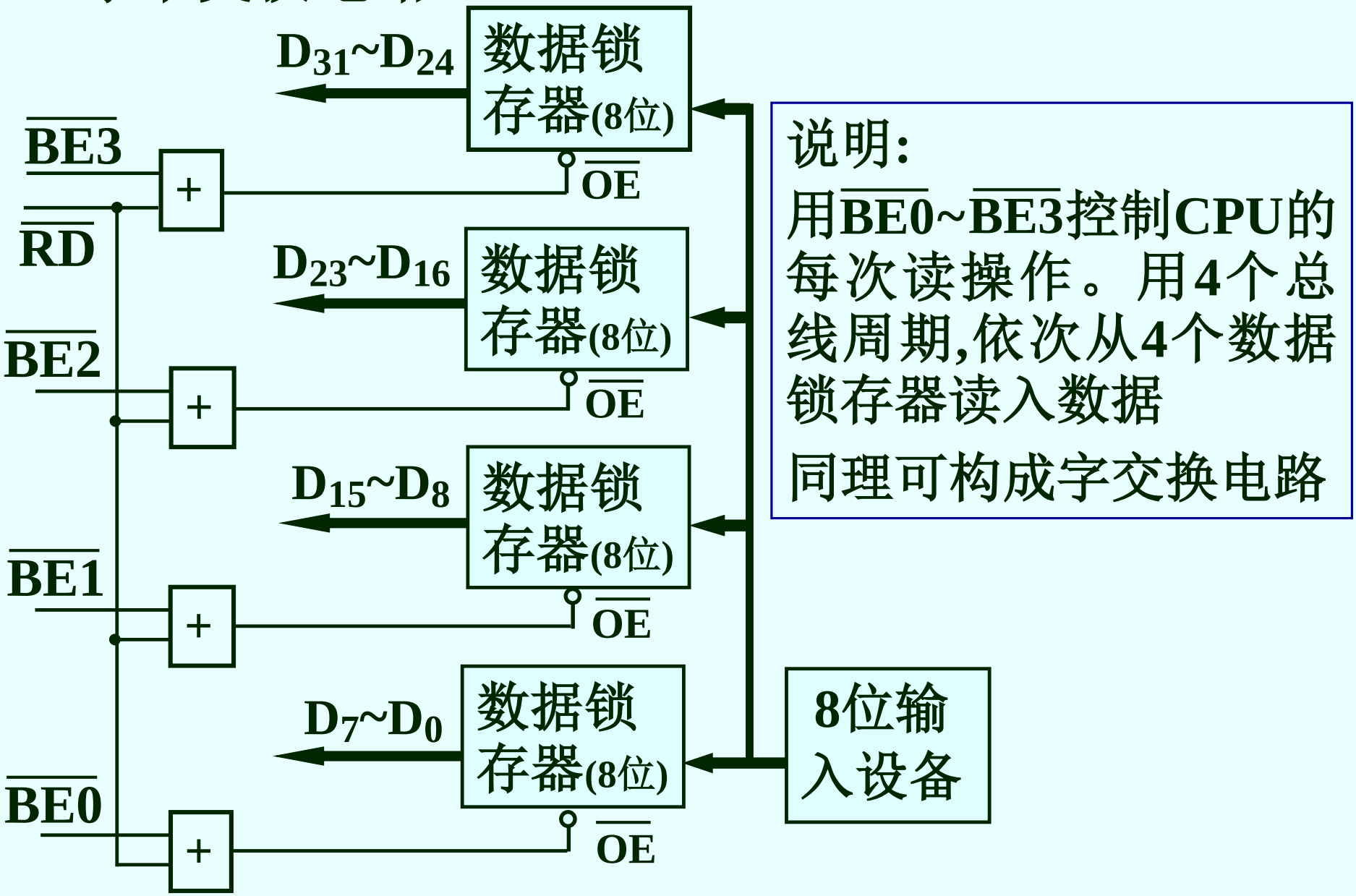
外设的8位数据线连接到系统数据总线的四个字节的哪个字节上?

解决方法:

用中间电路进行转换, 保证16位或8位外设的数据能依次传送到数据总线的不同部分, 如下图所示:



– 字节交换电路


D_{31}~D_{24}
数据锁存器(8位)
$\overline{OE}$
$\overline{BE3}$
$\overline{RD}$
+
D_{23}~D_{16}
数据锁存器(8位)
$\overline{OE}$
$\overline{BE2}$
+
D_{15}~D_{8}
数据锁存器(8位)
$\overline{OE}$
$\overline{BE1}$
+
D_{7}~D_{0}
数据锁存器(8位)
$\overline{OE}$
$\overline{BE0}$
+
8位输入设备
说明:
用$\overline{BE0}$~$\overline{BE3}$控制CPU的每次读操作。用4个总线周期,依次从4个数据锁存器读入数据
同理可构成字交换电路

4. 总线请求及总线响应信号

- **HOLD:** 总线请求信号 (总线主设备与从设备)
- **HLDA:** 总线响应信号
- **AHOLD:** 地址保护请求信号

一个总线主控设备向处理器提出的释放系统地址总线的请求信号。

该信号与 $\overline{\text{EADS}}$ 信号一起控制Cache操作。

5. Cache控制信号

- $\overline{\text{KEN}}$ 有效时, 可以填充Cache
- $\overline{\text{FLUSH}}$ 有效时, 清除Cache
- **PCD PWT** 用于控制片外Cache

- $\overline{EADS}$

$\overline{ADS}$ CPU*输出*信号, 指示CPU总线周期开始(地址及相应控制信号已发到总线上)。

$\overline{EADS}$ CPU的*输入*信号, 向CPU指示有其它主控设备已将要进行内存写入操作的地址放到了系统地址总线上。

用于80486总线监视地总线监视功能

便于在必要的情况下, 将Cache某些单元标识为无效。

$\overline{EADS}$与AHOLD一起, 构成一个“使无效”的周期,

工作过程如下:

① 某主控设备发出AHOLD→80486, 要求80486释放系统地址总线;

② 该设备发出地址有效信号→80486 (由$\overline{\text{EADS}}$接收) 表示该设备写入内存的地址已放到系统地址总线;

③ 80486读取该地址(80486地址总线为双向), 与自身Cache中已有的地址进行比较, 若该地址存在, 则标识该地址单元内容无效。

6. 总线仲裁信号

- **BREQ** CPU的输出信号, 用于多机系统的总线请求信号
- $\overline{\text{BOFF}}$ CPU收到该信号, 将被强制让出总线使用权

7. 总线状态信号 $\overline{RDY}$ $\overline{BRDY}$ $\overline{BLAST}$

$\overline{RDY}$ 与8086的Ready含义相同

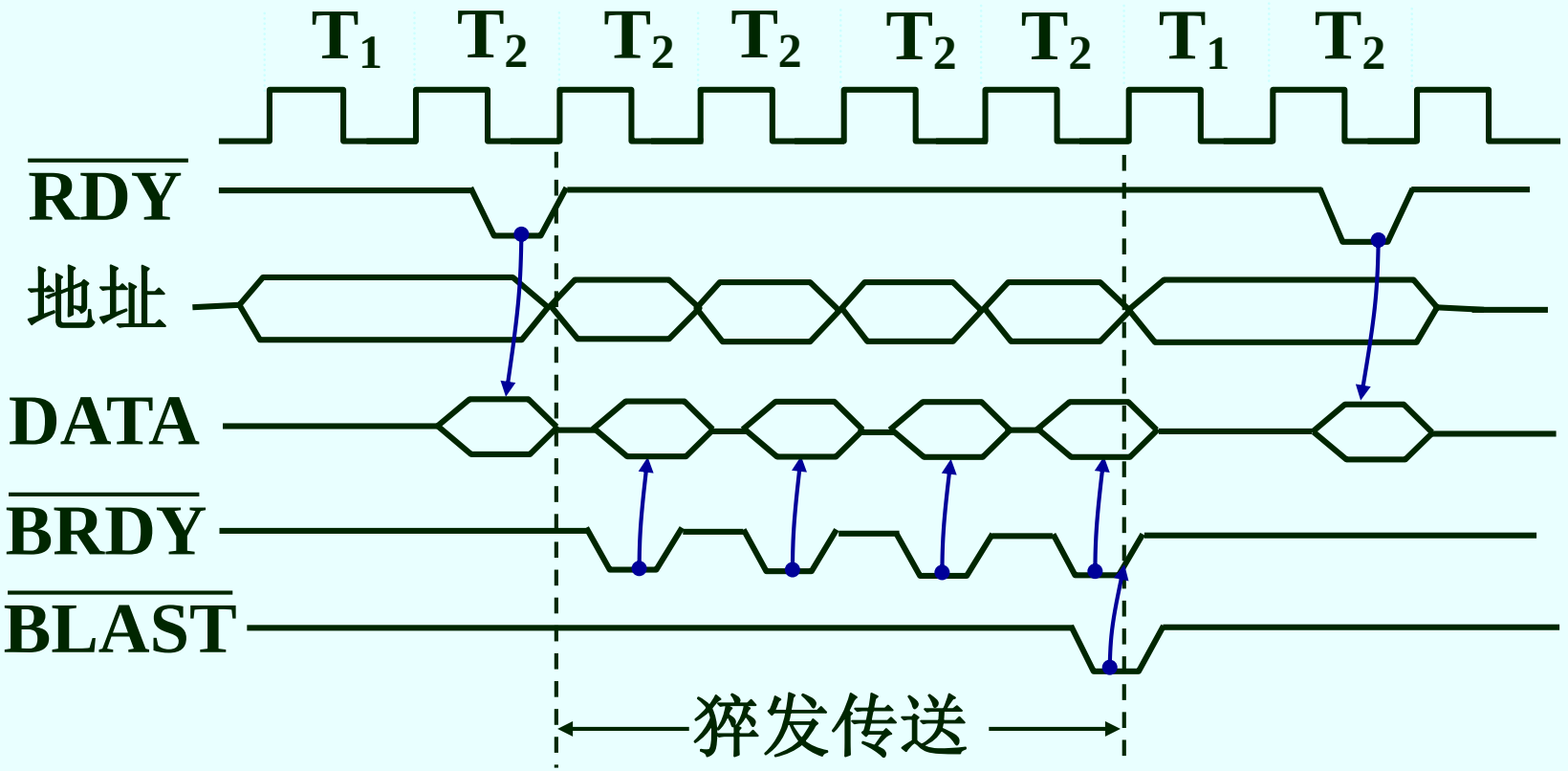
$\overline{BRDY}$ $\overline{BLAST}$ 用于猝发(Burst)传送

在80486中, 猝发传送是指每一个T(而不是每一个总线周期2个T)传送一个数据。

猝发传送由外设向CPU提出请求后进行, 外设发送信号 $\overline{BRDY}$ (“猝发传输准备好”)来实现。

每个时钟节拍T, 让 $\overline{BRDY}$ 有效, CPU在每个T采样数据总线。

如下图所示:



结束猝发传送有两种方式:

- 被访问设备主动结束: 停止$\overline{\text{BRDY}}$有效;
- CPU主动中止: 发出$\overline{\text{BLAST}}$信号。