第三节 Intel 80386 微处理器

Intel 80386简述

1. 32位的总线宽度, 支持4000M的存储空间;
2. 一定程度上的指令流水线;
3. 双重的虚地址保护功能(分段和分页保护);
4. 减少了每个总线周期的T时钟数;
5. 支持数据总线的8、16、32位数据传送;
6. 支持片外高度缓存。

一、80386的内部结构

(一) 功能部件(六大功能部件)

(1) BIU(总线接口部件):

完成指令预取请求和执行单元的数据存取请求,数据存取请求优先于指令预取请求。

(2) IPU(指令预取部件):

16字节指令预取队列，提出预取请求

(3) IDU(指令译码部件):完成指令译码。

(4) EU(执行部件):

完成指令所要求的功能。流水线功能表现为:一条指令的执行与下一条指令的取指操作并行

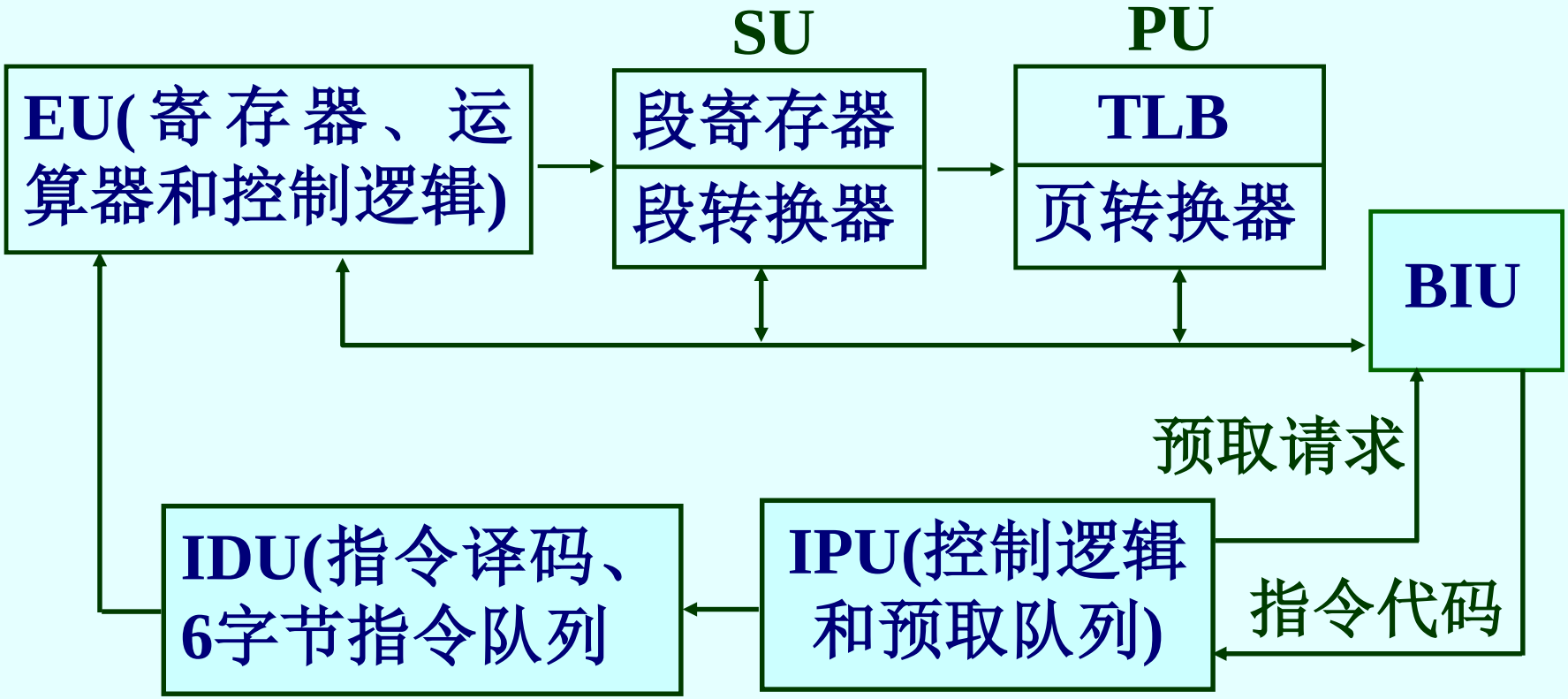
(5) SU(分段部件):

完成执行单元的地址请求, 将虚地址转换为线性地址

线性地址{不允许分页: 线性地址=物理地址
若允许分页: 线性地址为页地址(虚地址)

(6) PU(分页部件): 将线性地址转换为物理地址

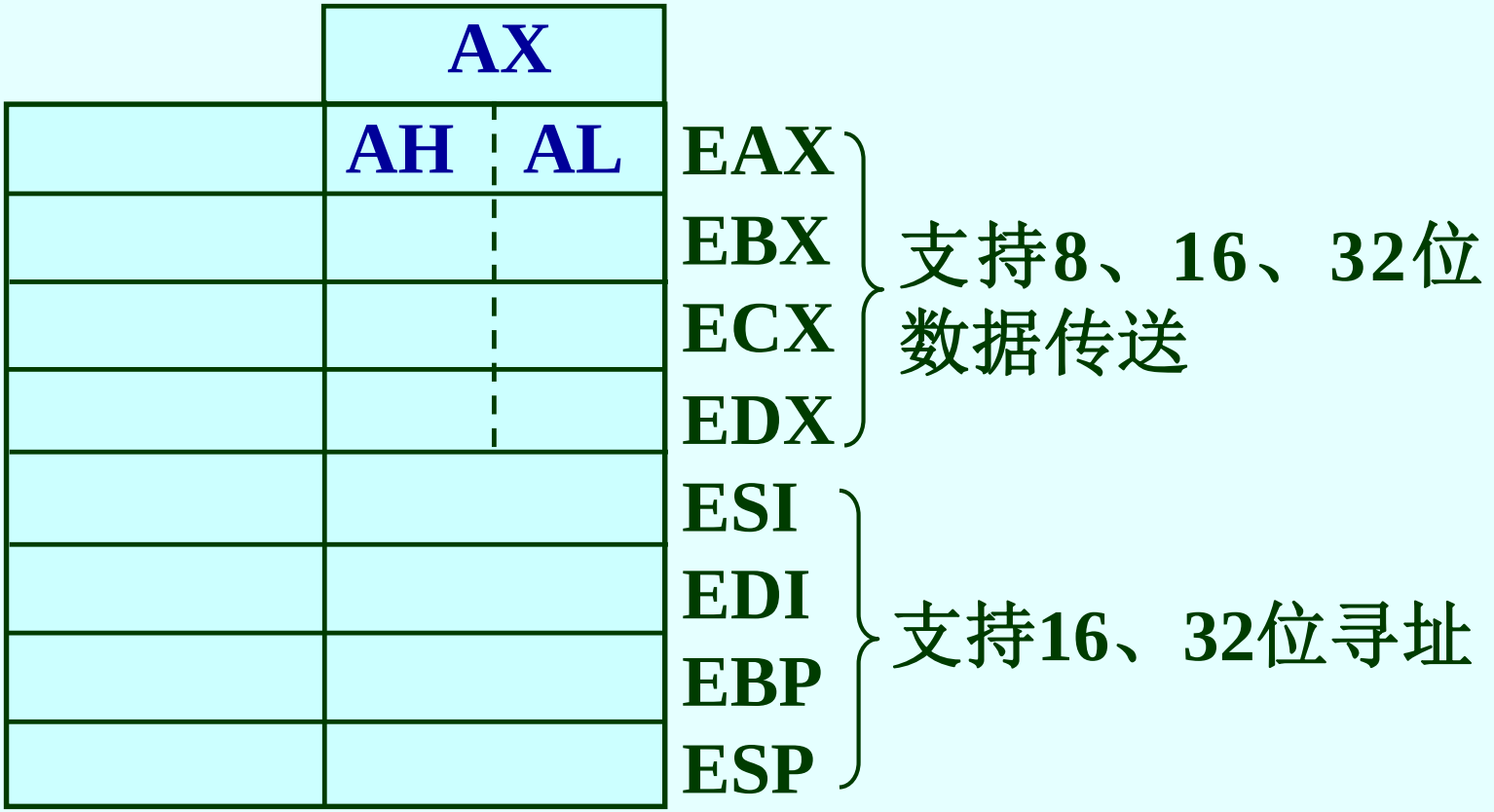
各功能部件的逻辑关系如下图所示:



(二) 80386寄存器

按功能, 寄存器可以分为6个大类:

1、通用寄存器(8个)

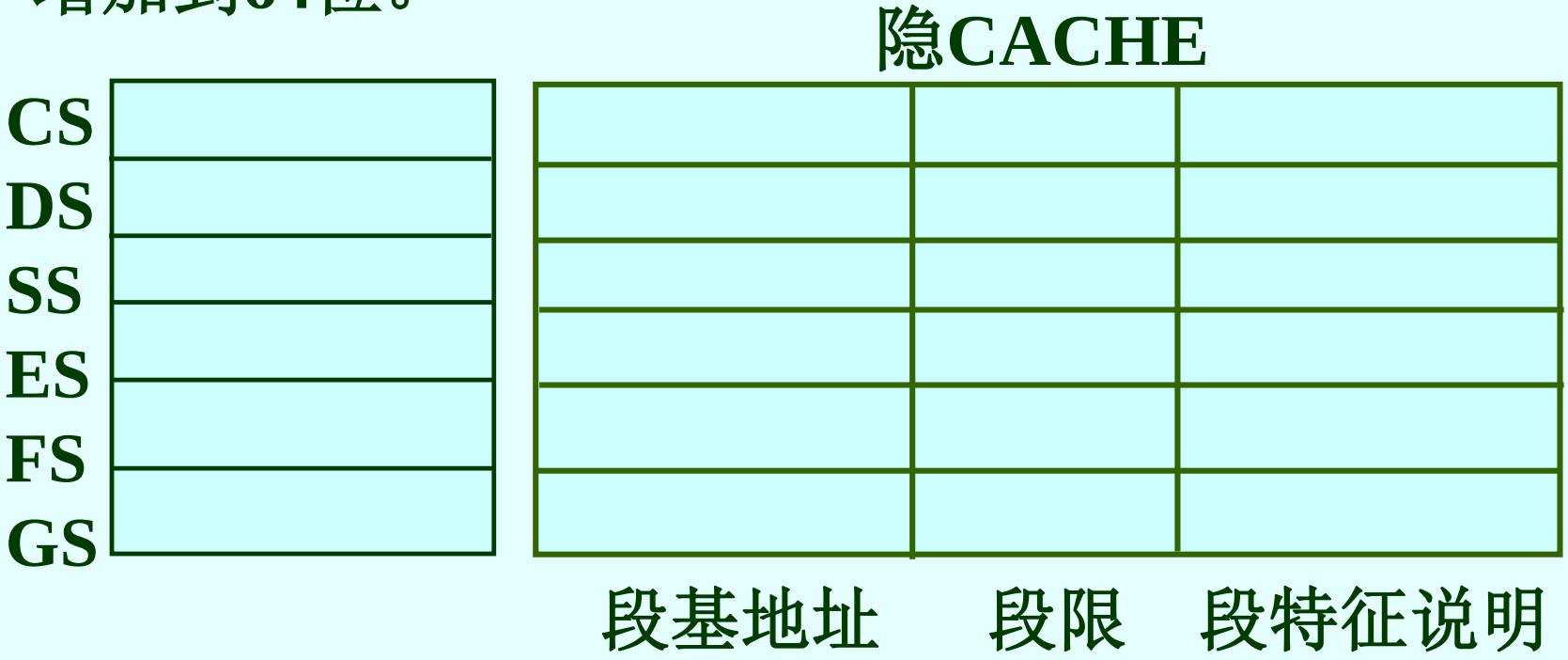


2、段寄存器(6个)

CS、DS、SS、ES与80286相同

增加了两个数据段寄存器FS和GS, 以支持更多的数据类型(如整数、浮点数、串等)。

80386的描述子的8个字节都有意义, 隐Cache的长度增加到64位。



3、控制寄存器(4个) CR_0～CR_3

(1) CR_0:

由80286的MSW寄存器演变而来，并增加了2位

- PG { 1: 允许分页 / 0: 不允许分页
- ET: 协处理器扩展位 { 1: 系统配置80387 / 0:系统配置80287或无协处理器

(2) CR_1: 未使用

(3) CR_2: 页故障地址寄存器, 存放出现故障的页的32位线性地址

(4) CR_3: 页目录基地址寄存器, 存放页目录表的基地址

二、80386部分引脚功能

1、数据引脚: D_{31}～D_0

2、地址总线32位: A_{31}～A_2 $\overline{BE}_3$ $\overline{BE}_2$ $\overline{BE}_1$ $\overline{BE}_0$

字节选择信号

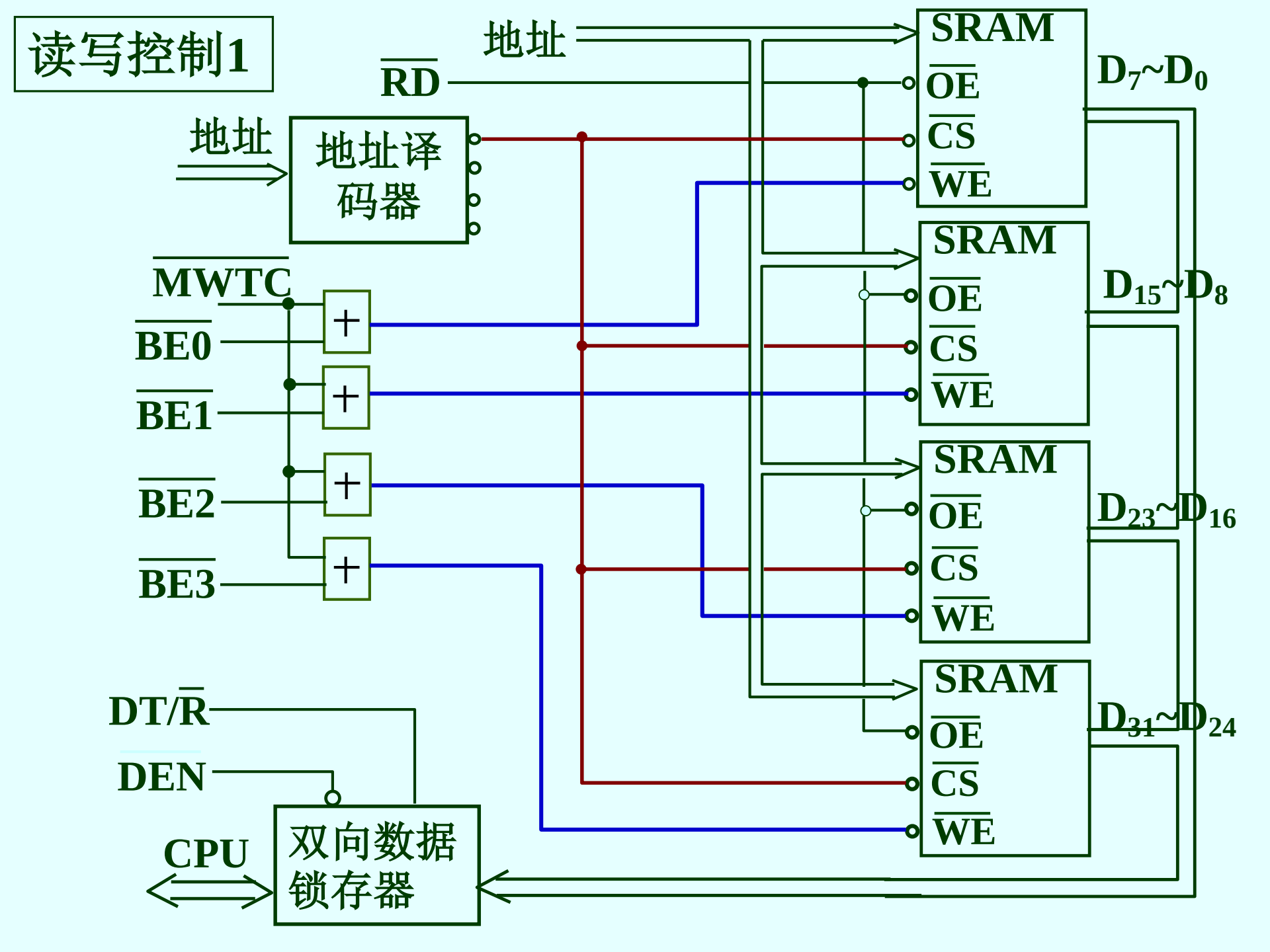
当 $\overline{BE}_3$ 有效时: 选择 D_{31}～D_{24} (等效于A_1A_0=11)

$\overline{BE}_2$ 有效时: 选择 D_{23}～D_{16} (等效于A_1A_0 =10)

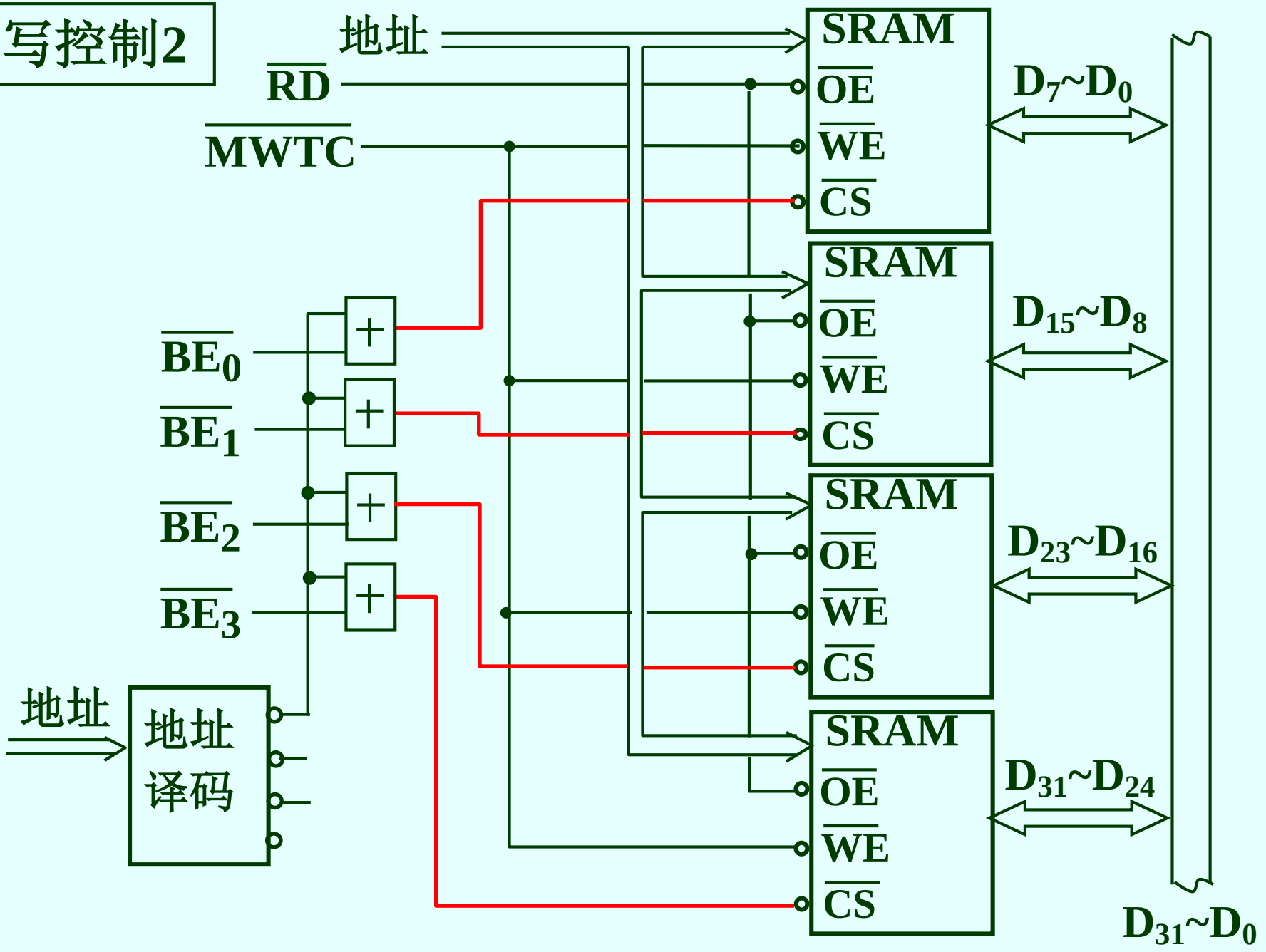
$\overline{BE}_1$ 有效时: 选择 D_{15}～D_8 (等效于A_1A_0 =01)

$\overline{BE}_0$ 有效时: 选择 D_7～D_0 (等效于A_1A_0 =00)

$\overline{BE}_3$~ $\overline{BE}_0$ 线路连接例: CPU与SRAM的连接框图


读写控制1
地址
RD
地址
地址译码器
MWTC
BE0
BE1
BE2
BE3
SRAM
OE
CS
WE
D7~D0
SRAM
OE
CS
WE
D15~D8
SRAM
OE
CS
WE
D23~D16
SRAM
OE
CS
WE
D31~D24
DT/R
DEN
CPU
双向数据锁存器

读写控制2


地址
RD
MWTC
SRAM
OE
WE
CS
D7~D0
D15~D8
D23~D16
D31~D24
D31~D0
BE0
BE1
BE2
BE3
地址
译码

问题: 为什么80386以及80486的地址引脚设置成

A_{31}～A_2 $\overline{BE_3}$ $\overline{BE_2}$ $\overline{BE_1}$ $\overline{BE_0}$

而不是 A_{31}～A_2 A_1 A_0 ?

思路: 如何实现?

同时选中 四个字节

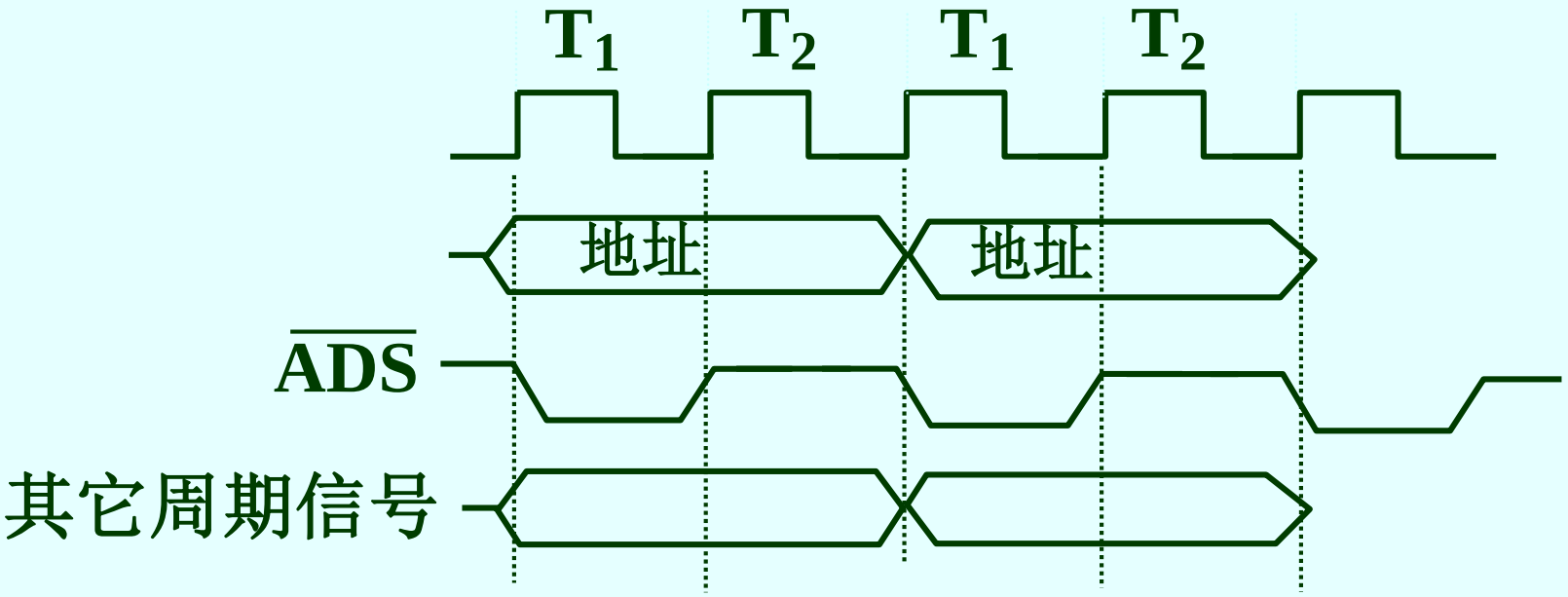
或者只选 三个字节

或者只选 二个字节

或者只选 一个字节

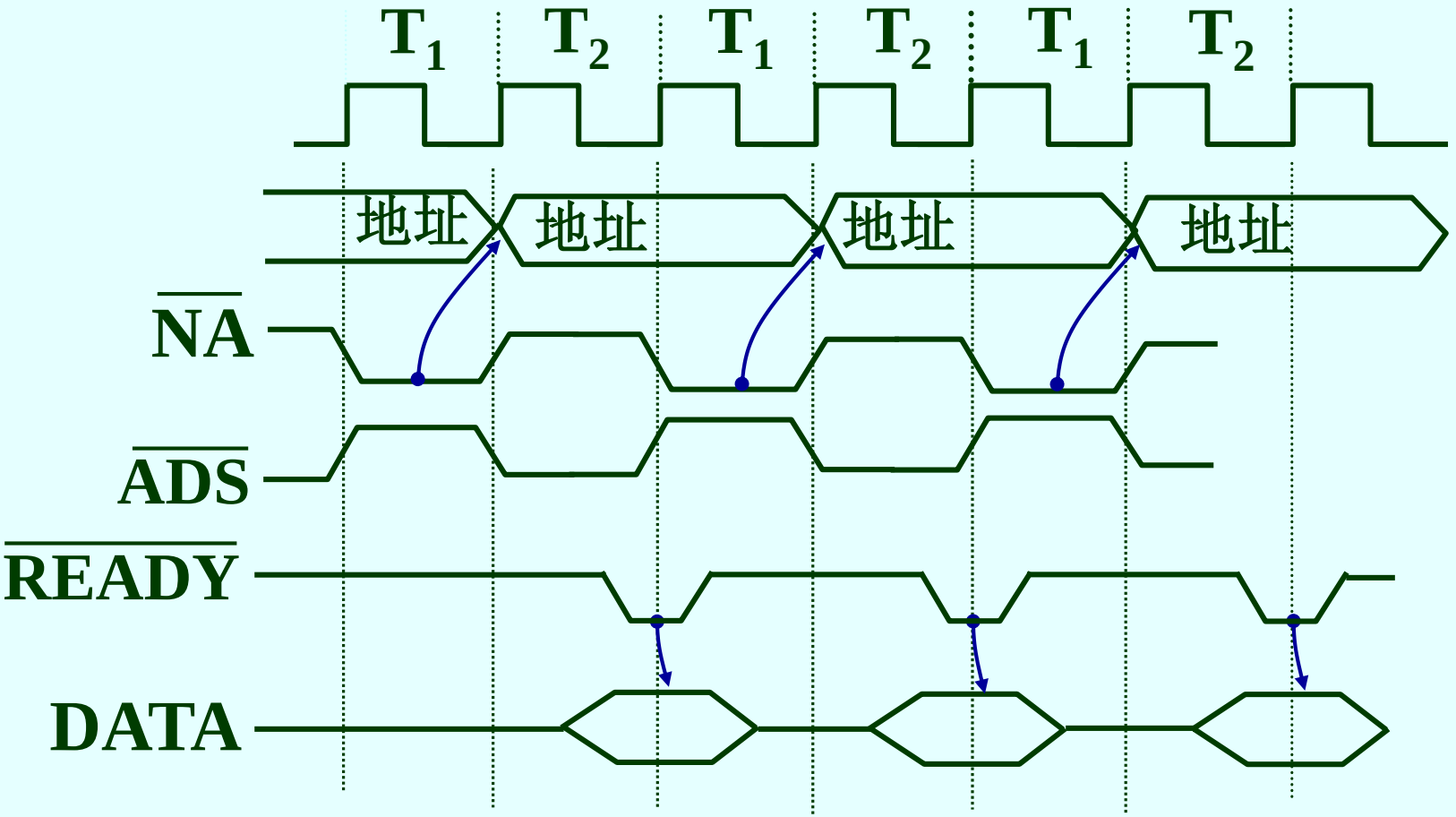
3、总线控制信号（$\overline{ADS}$ $\overline{READY}$ $\overline{NA}$ $\overline{BS16}$）

① $\overline{ADS}$ 地址状态信号



② $\overline{NA}$ 下一个地址请求

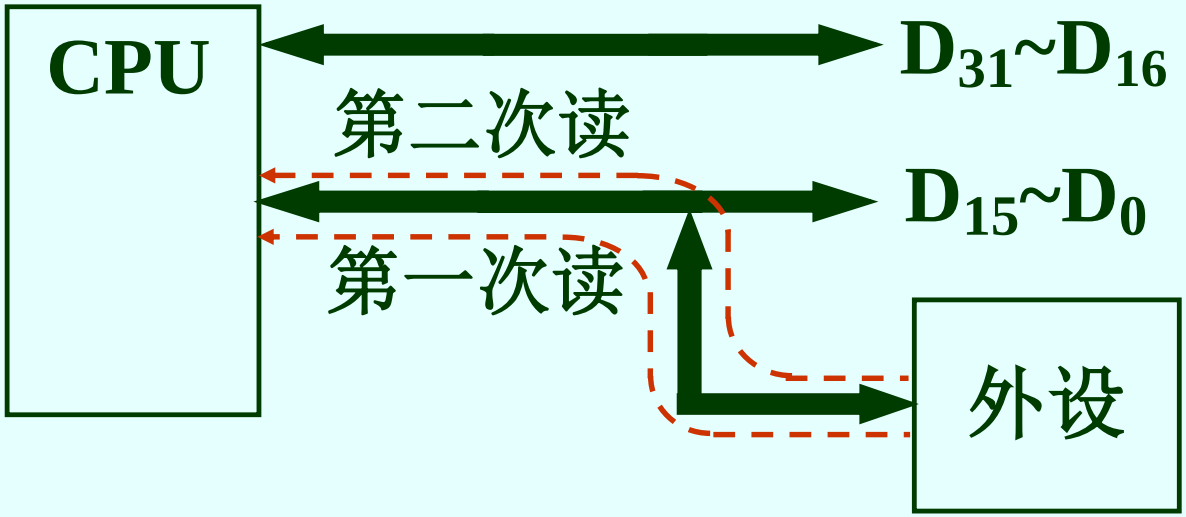
作用：请求CPU进入地址流水线方式



③ $\overline{BS16}$ 总线宽度控制信号

请求(强制)CPU只使用数据总线的低16位

- 读周期: 只采样数据总线的低16位 D_{15}~D_0
  如果CPU执行32位读操作，则CPU自动执行2个16位的读操作周期。



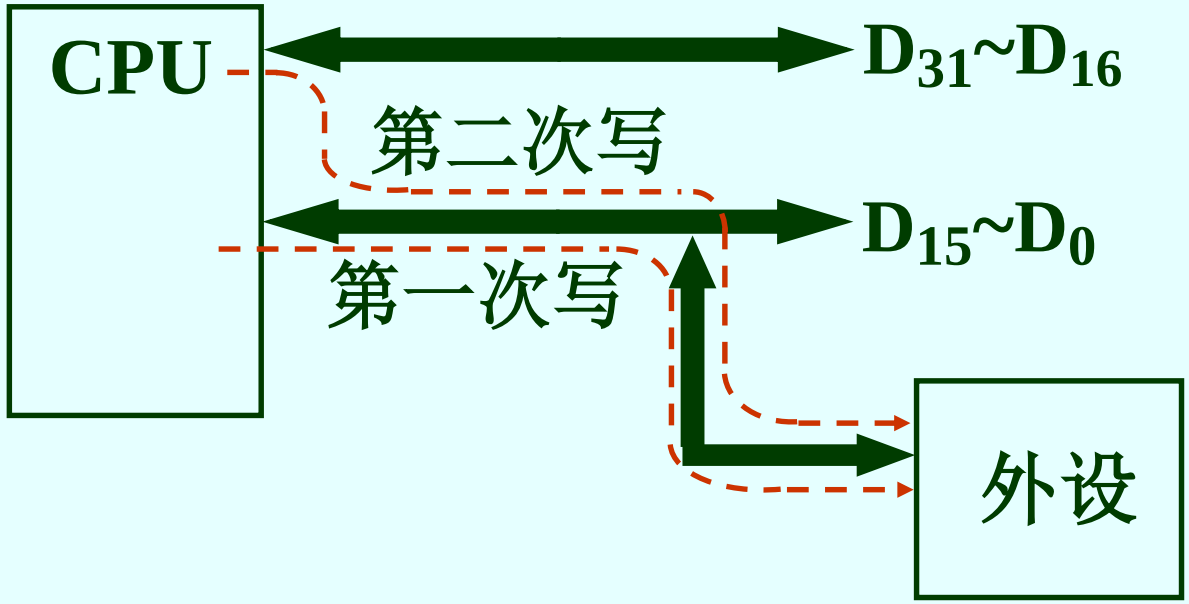
因此, 被读取的设备只将数据发送到D_{15}~D_0。

- 对写周期

  CPU只将数据写到D_{15}~D_0上。

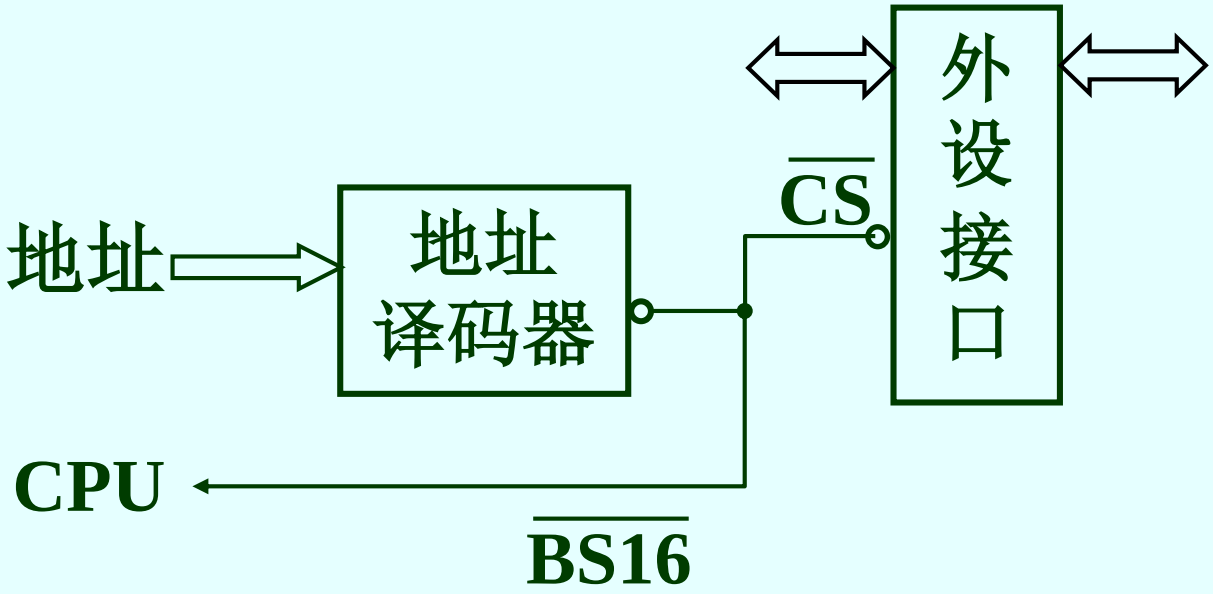
  如果CPU执行32位的写操作，则CPU自动执行2个16位的写操作周期。

  CPU将本应在D_{31}~D_{16}上传送的数据复制到D_{15}~D_0上传送。



— $\overline{\text{BS16}}$的来源:

原理上, 该信号来自于地址译码



三、80386的保护模式

(一) 实地址模式

通电或RESET时, PE位置0, 进入实地址模式。

(二) 保护模式

80386提供分段和分页两种存储管理模式

- 分段管理的特点: 对存储空间的逻辑划分

  优点: 对模块化(结构化)程序设计方式提供了很好的支持

  缺点: ① 段长可变不固定, 管理更复杂;
  ② 空间碎片较多;
  ③ 访问字节数/传输字节 (比值)可能很小

- 分页管理的特点: 对存储空间的物理划分

优点: ① 页面长度固定, 管理机构简单

② 访问字节数/传输字节 (比值)通常更大

缺点: ① 一个页面内的程序或数据可能不具有完整意义, 不利于结构化程序设计

② 存在“内零头”

提供两种存储管理方式, 吸收两种模式的优点, 可为不同的操作系统提供不同的硬件支持。

80386的分页机制采用每4K为一个页面。对页面的寻址需要提供页面基地址和页内偏移量。

1、分段管理

80386的数据代码段描述子的格式:

7	段基地址$_{31\sim24}$	G	D/B	0	0	段限$_{19\sim16}$	6
5	访问权字节	段基地址$_{23\sim16}$					4
3	段基地址$_{15\sim0}$						2
1	段限$_{15\sim0}$						0

- G: 粒度位
  - 0: 段长以字节为单位
  - 1: 段长以页面为单位
- D/B:
  - 0: 按16位操作, 与80286兼容
  - 1: 按32位操作
- 访问权字节

P	DPL	S	E	ED/C	W/R	A

与80286相同

2、分页管理

CR_0中的PG位=0, 不允许分页，经分段部件出来的地址(即线性地址=物理地址);

如果PG=1, 允许分页, 经分段部件产生的线性地址仍为虚地址, 须经分页部件转换成物理地址。

(1) 分页的寻址过程

从虚地址到实地址的转换过程

目标： 根据线性地址, 得到物理页面基地址以及该页面内的偏移量

假设：基于简单查表的转换机构, 用虚拟页号查表, 表中内容为实际页面的基地址

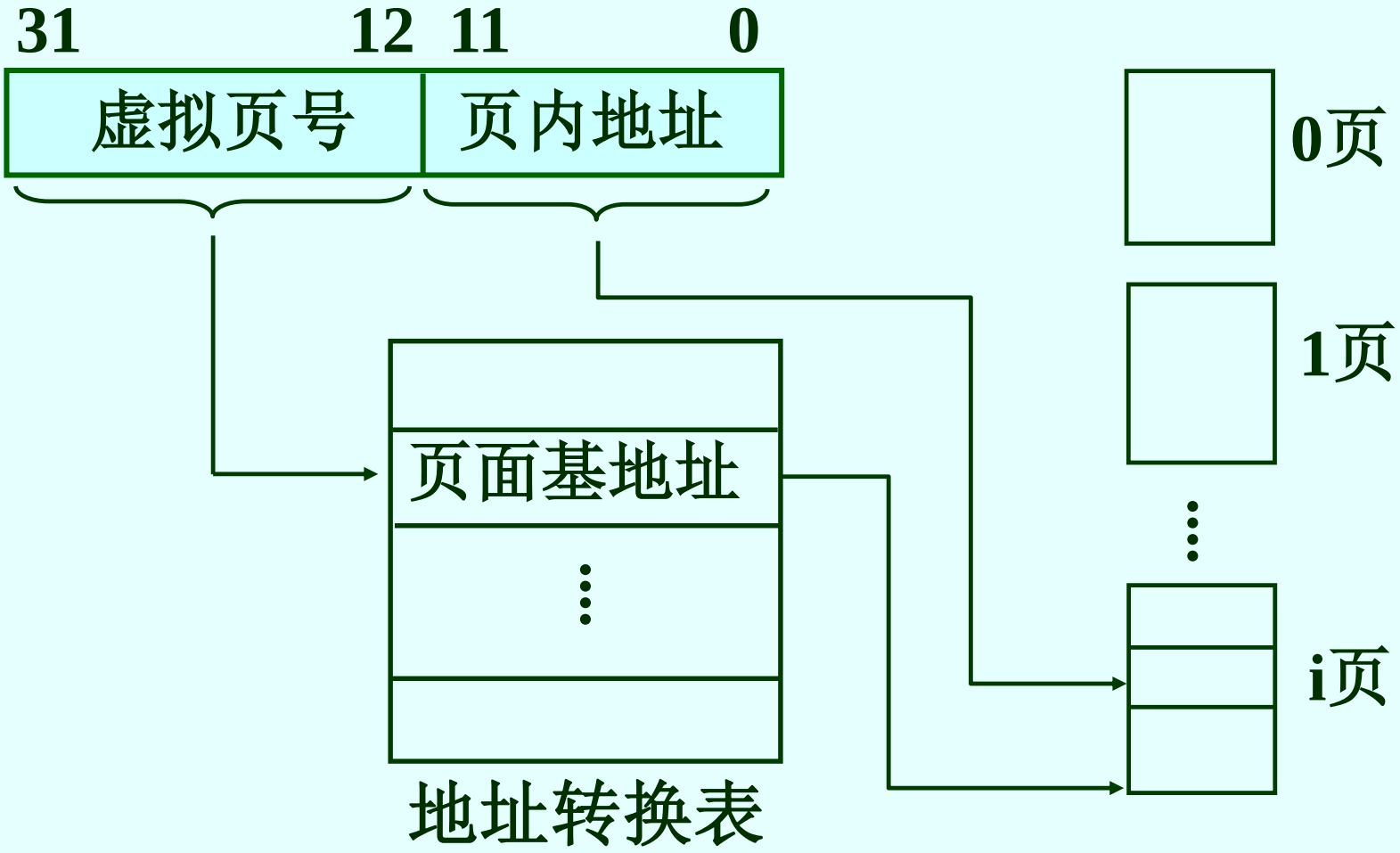
根据每4K为一个页面, 4000M分为1M个页面。

将32位的线性地址分为两个部分, 即高20位(2^{20} =1M)和低12位(作偏移量)。高20位作为虚拟页号。

为此, 建立一张与虚拟页号对应的实际页面的基地址表, 20位可查询1M个表项, 表中内容为实际页面的基地址, 该基地址与线性地址12位(偏移量)得到访问单元的实地址。

如下图所示:

分段产生的线性地址



基于简单查表的转换机构的不可行性

二级查表机制(实际采用):

- 第一级表:

  称为页目录, 共1024项(称为页目录项)。每一项用来指示一个二级表(的基地址);

- 第二级表:

  称为页表, 共1024项(称为页表项)。每一项用来指示一个页面的基地址。

由此可得: 二级表共可以指示1024×1024个页面的基地址, 即1M个页面的基地址。

页目录只有一个, 大小为4K, 常驻内存; 页表1024个, 不常驻内存。

二级表的地址转换过程:

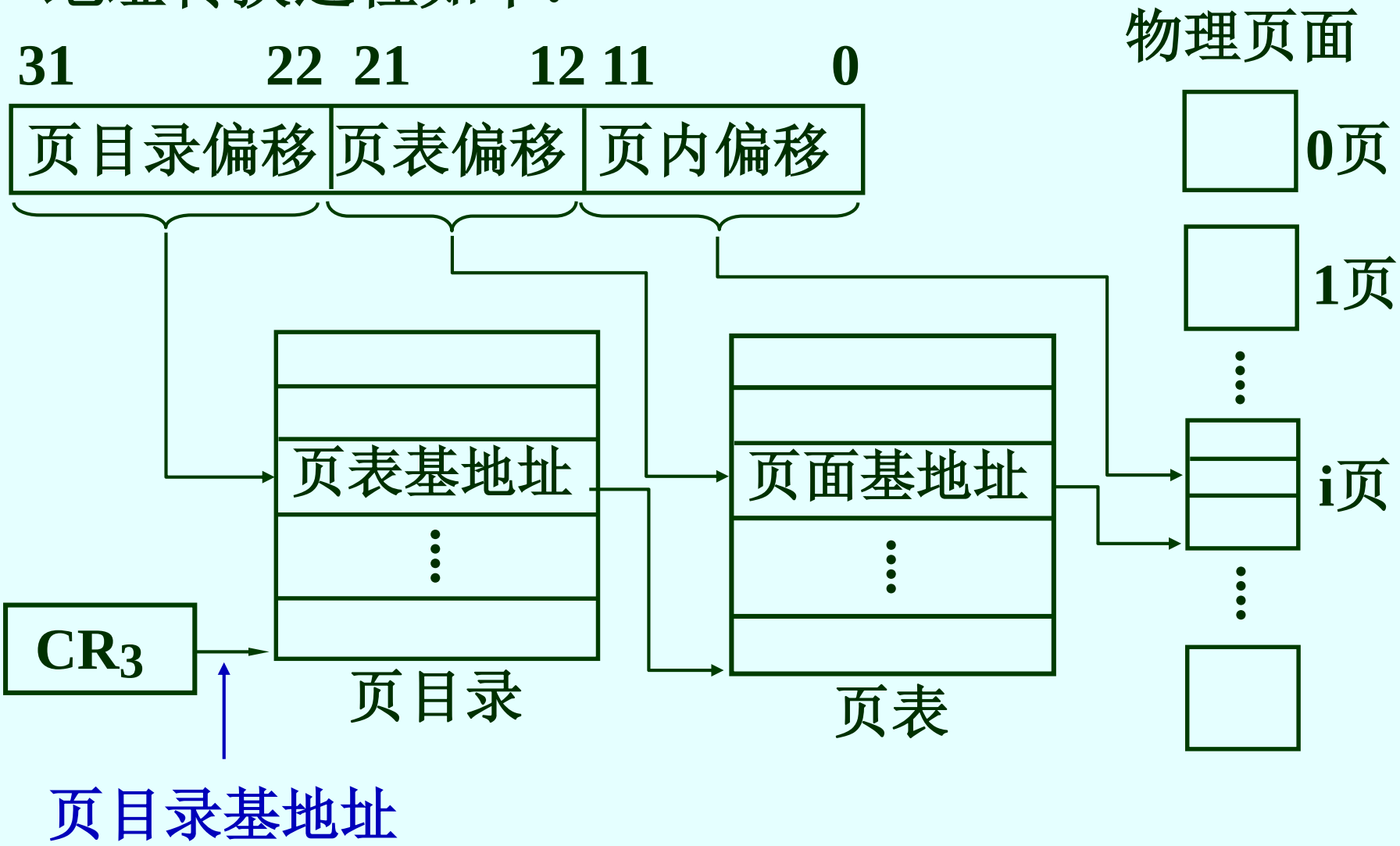
二级表的地址转换过程:

将虚拟地址的高20位分成两部分, 分别用于指示两种不同表的表项(作为表内的偏移量)。

如下图所示:

- D_{31}～D_{22}指示1024个目录项的某一项, 该项的内容为一个页表的基地址;
- D_{21}～D_{12}指示1024个页表项的某一项, 该项的内容为页面的基地址;
- D_{11}～D_0指示页面的偏移量。

地址转换过程如下：


物理页面
31
22
21
12
11
0
页目录偏移
页表偏移
页内偏移
0页
1页
i页
页表基地址
页面基地址
CR3
页目录
页表
页目录基地址

(2) 页目录项和页表项的组成

- 页目录项

31 12	11 9	8	7	6	5	4	3	2	1	0
页表基地址	OS专用	0	0	D	A	×	×	U/S	R/W	P

- 页表项

31 12	11 9	8	7	6	5	4	3	2	1	0
页面基地址	OS专用	0	0	D	A	×	×	U/S	R/W	P

为什么页表基地址和页面基地址只有20位？

- 页的长度4K, 基地址单位量的增/减跨越4K空间
- 对齐的观点

控制位:

- D 修改位
  - 页目录项的D位无意义;
  - 页表项D位
    - 1 表项对应的页面在内存中已被修改, 在换页时, 需要将内存中的该页面先写回磁盘
    - 0 未作修改, 换页时不必写回磁盘

- A 访问位
  - 页目录项A位
    - 1 对应的页表已被访问过
    - 0 对应的页表未被访问过
  - 页表项A位
    - 1 对应的页面已被访问过
    - 0 对应的页面未被访问过

操作系统可用该位并结合时钟对页面实施LRU淘汰算法

操作系统可用该位并结合时钟对页表实施LRU淘汰算法

31 12	11 10 9	8	7	6	5	4	3	2	1	0
页表/页面基地址	OS专用	0	0	D	A	×	×	U/S	R/W	P

- **P** 存在位
  - 1 对应的页表或页面在内存空间
  - 0 对应的页表或页面不在内存空间

- U/S R/W

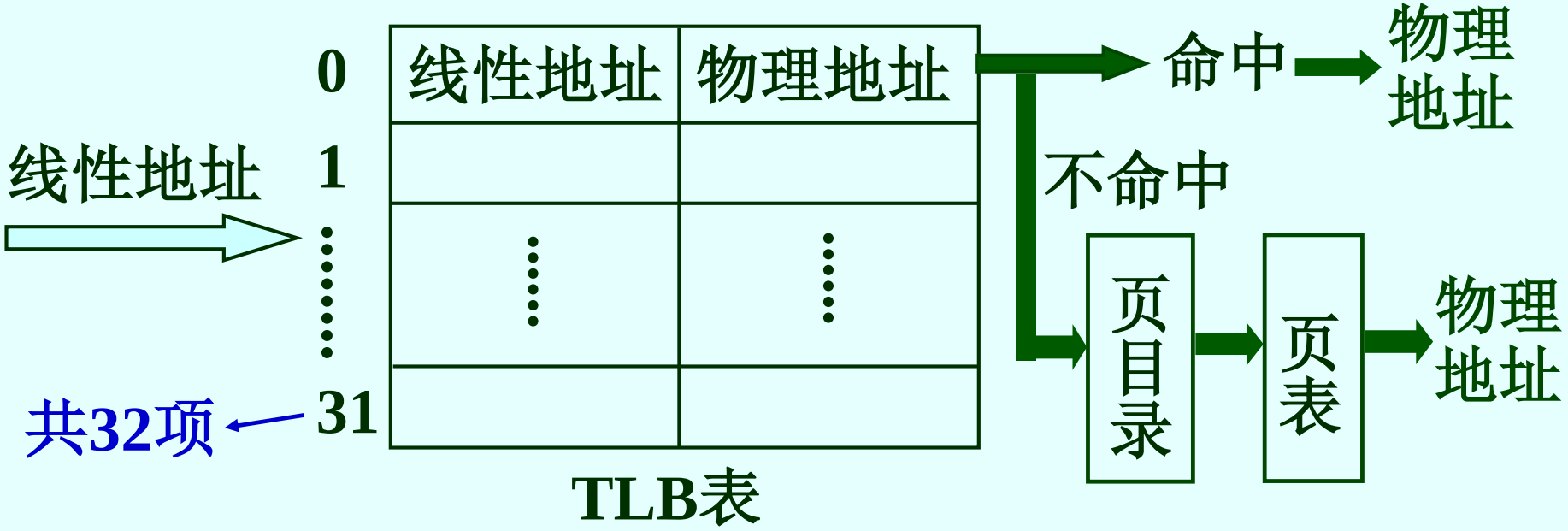
U/S	R/W	CPL=3	CPL=0、1、2
0	0	不允许访问	可读/可写
0	1	不允许访问	可读/可写
1	0	可读、不可写	可读/可写
1	1	可读、可写	可读/可写

(3) TLB表(Translation Lookaside Buffer, 快表)

TLB用于加快页地址转换的Cache

① 工作原理

将最近访问的页的物理地址存放于TLB



如果用线性地址对TLB表的32个表项进行线性查找, 则平均查找次数为16次, 显然会导致页面访问速度很低。

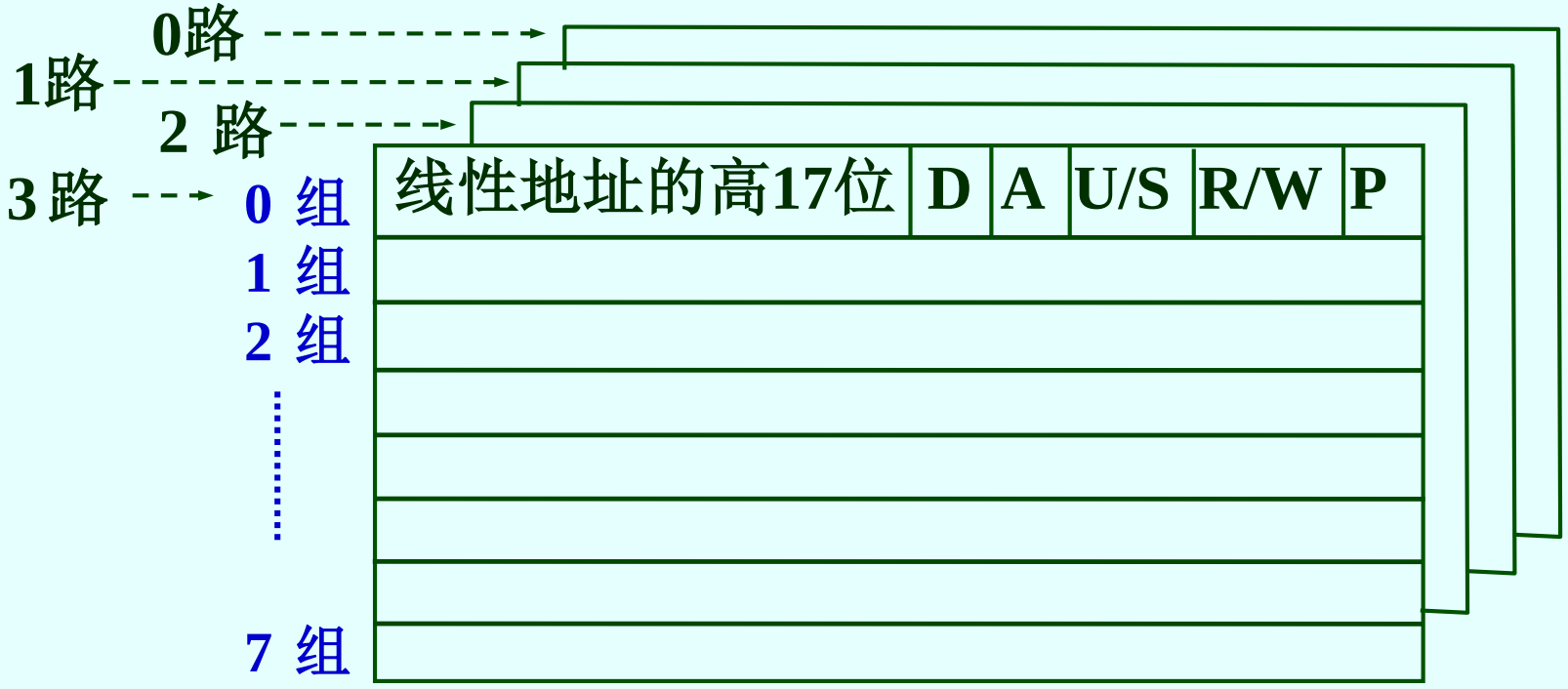
② TLB表的结构

– TLB表的每一项组成:

线性地址的高17位	物理页号	D	A	U/S	R/W	P

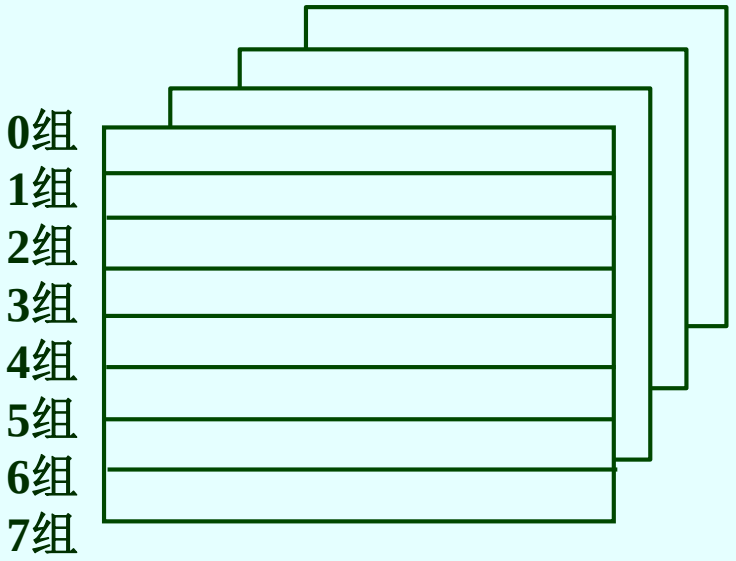
即页面基地址

32个表项构成一个4路8组的矩阵:



转换过程描述:

线性地址的高17位	物理页号	D	A	U/S	R/W	P



转换过程描述：

- 用线性地址高20位的低3位($D_{14}D_{13}D_{12}$)经译码选中某一组(比如5组);
- 用高17位与4路的5组(有4个5组)中所含的线性地址的高17位同时匹配(用比较电路可以实现), 哪一路匹配, 就选中那一路的5组;
- 被选中的组送出物理基地址并与线性地址的低12位(即页内偏移) 形成物理地址。