第九讲 存储系统

存储系统构成以及内存管理

华中科技大学
电子信息与通信学院
School of Electronic Information and Communications

学习目标

- 了解计算机系统存储系统的分级结构特点
- 内存的组织管理方式
  - 分页式
  - 分段式
  - 段页式
- 逻辑地址->物理地址
- 多字节访问效率

4.1计算机存储系统构成

CPU-Z

处理器 | 缓存 | 主板 | 内存 | SPD | 显卡 | 关于

处理器

名字	AMD Phenom X4 9650		
代号	Agena	TDP	95.9 W
插槽	Socket AM2+ (940)		
工艺	65 纳米	核心电压	1.104 V
规格	AMD Phenom(tm) 9650 Quad-Core Processor		
系列	F	型号	2
步进	3		
扩展系列	10	扩展型号	2
修订	DR-B3		
指令集	MMX(+), 3DNow!(+), SSE, SSE2, SSE3, SSE4A, x86-64, AMD-V		

时钟 (核心 #0)

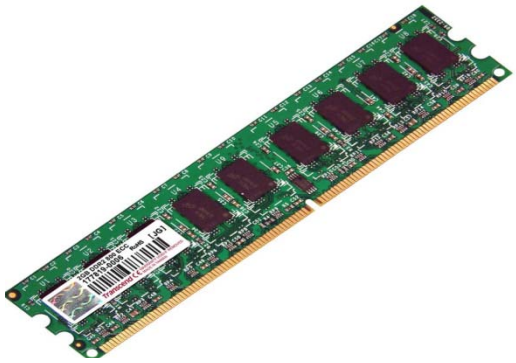
核心速度	1149.94 MHz
倍频	5.75 (5.75 - 11.50
总线速度	199.99 MHz
前段总线	1799.90 MHz

缓存

一级 数据	4 x 64 KBytes	2-way
一级 指令	4 x 64 KBytes	2-way
二级	4 x 512 KBytes	16-way
三级	2 MBytes	32-way

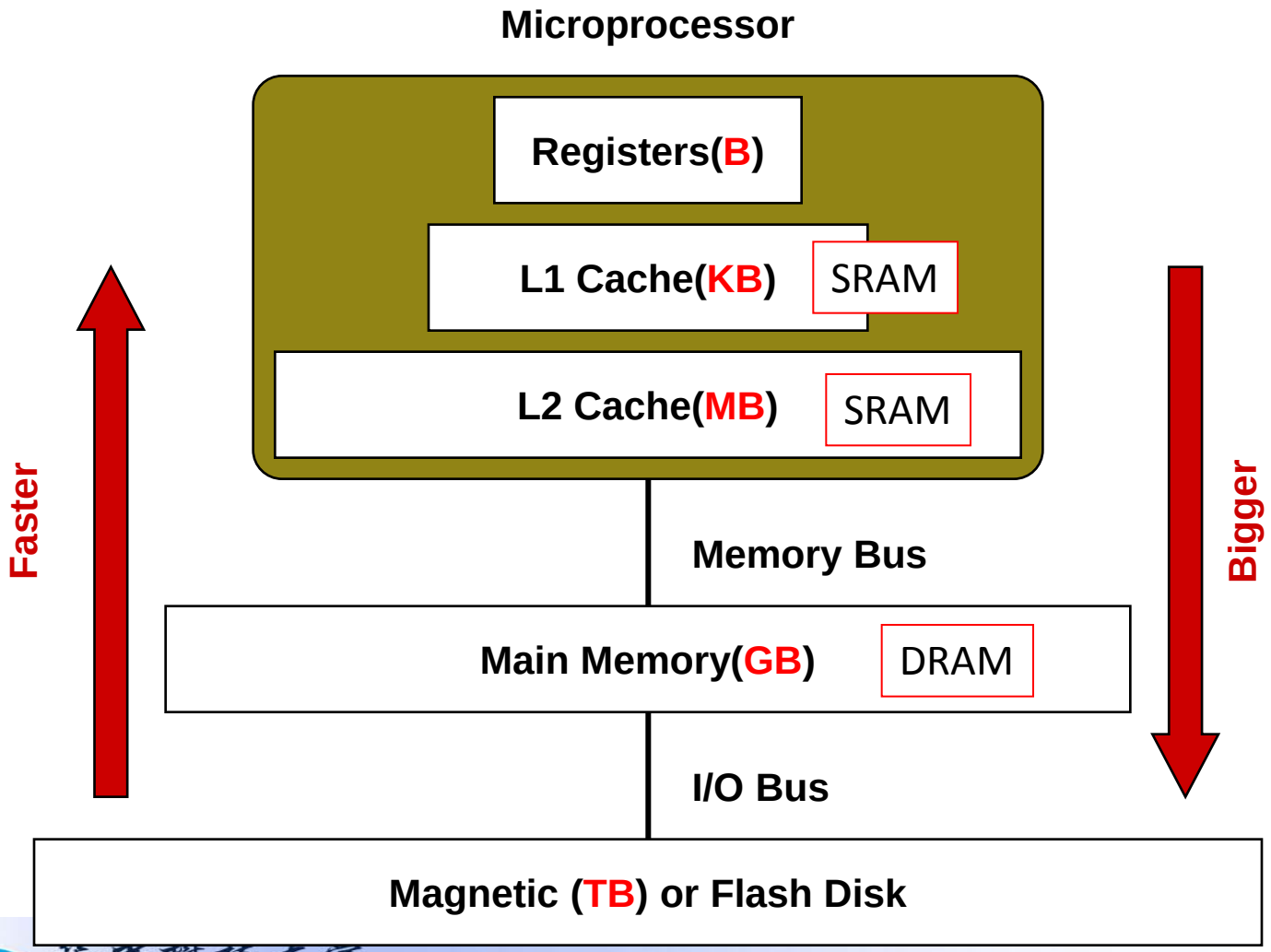
已选择 处理器 #1　核心数 4　线程数 4

CPU-Z Ver. 1.72.0.x32　工具　验证　确定



MyPrice

Typical Memory Hierarchy



Memory Technology Review
——capacity, speed, cost

- SRAM
  - <100MB
  - 0.5~2.5ns
  - 12000~30000 RMB per GB
- DRAM
  - <1 100GB
  - 10~50ns
  - 120~380 RMB per GB
- Magnetic Disk
  - >1TB
  - 5~20ms
  - 0.6~3.0 RMB per GB

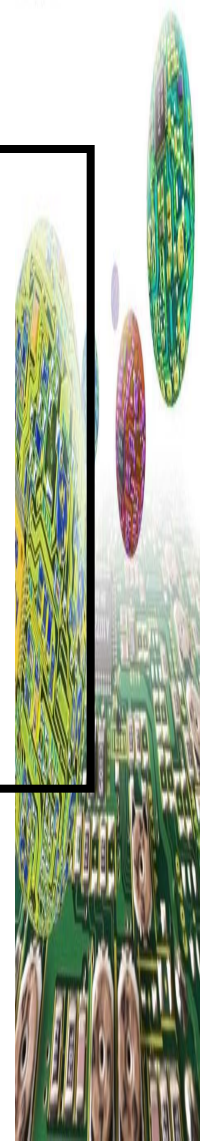
C. Can Memory Hierarchy improves the Performance

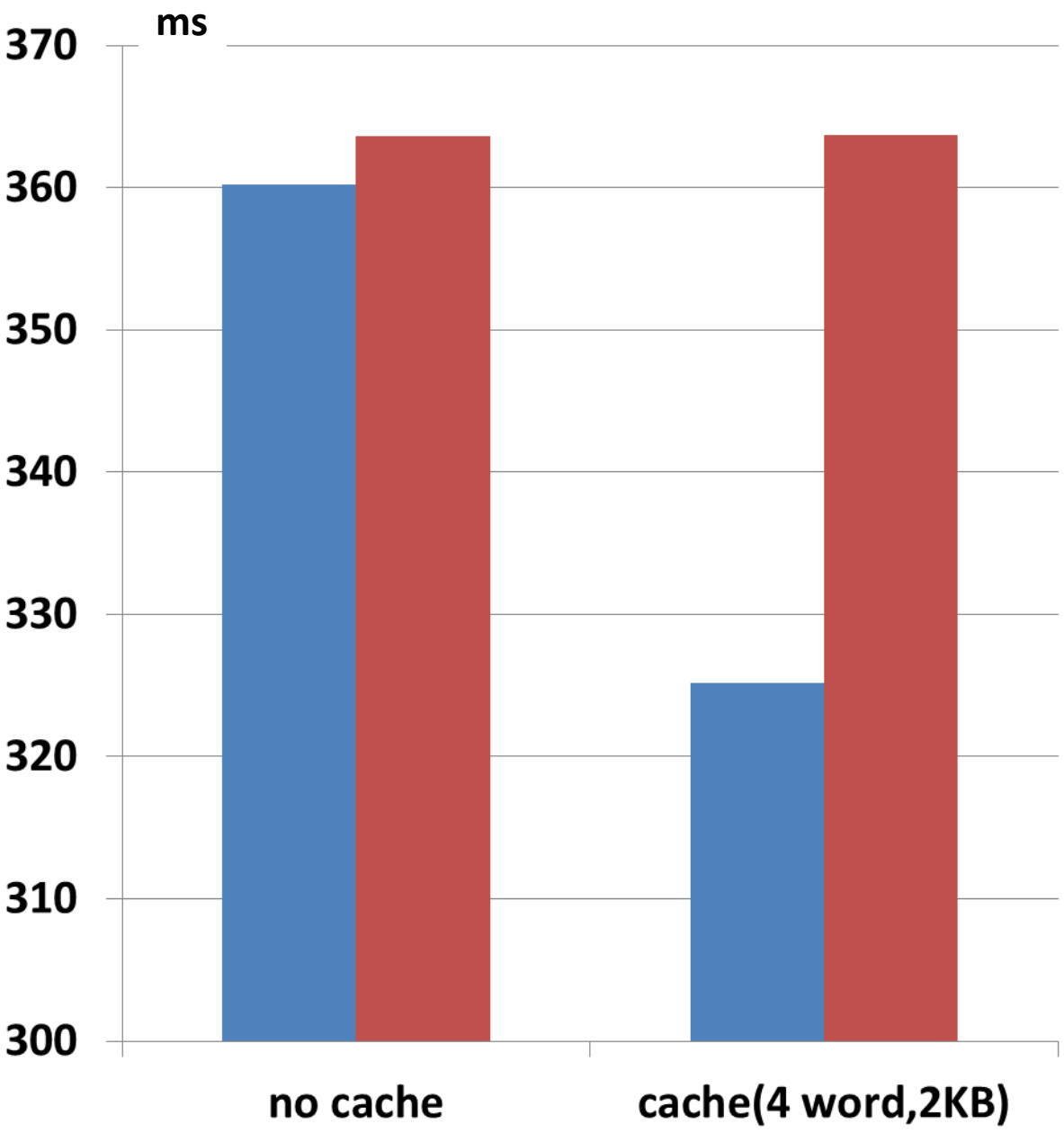
```
assign-array-rows()
{
 ……
  for  (i=0; i<M; i++)
          for (j=0; j<N; j++)
          sum= sum +a[i][j];
……
}
```

J first

```
assign-array-rows()
{
……
  for (j=0; j<N;j++)
          for  (i=0; i<M; i++)
          sum= sum +a[i][j];
……
}
```

I first

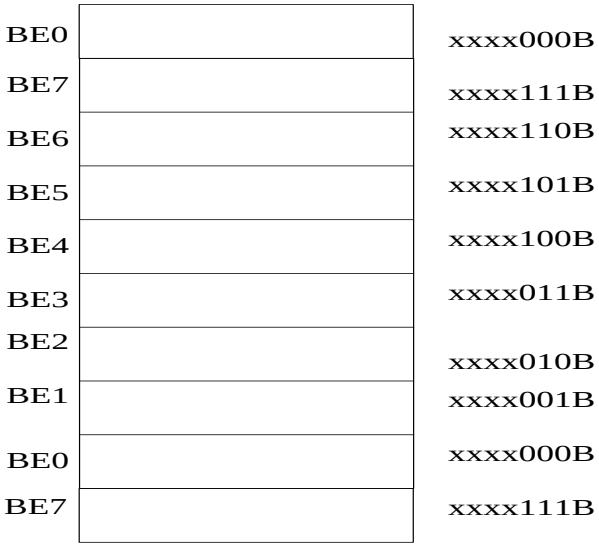




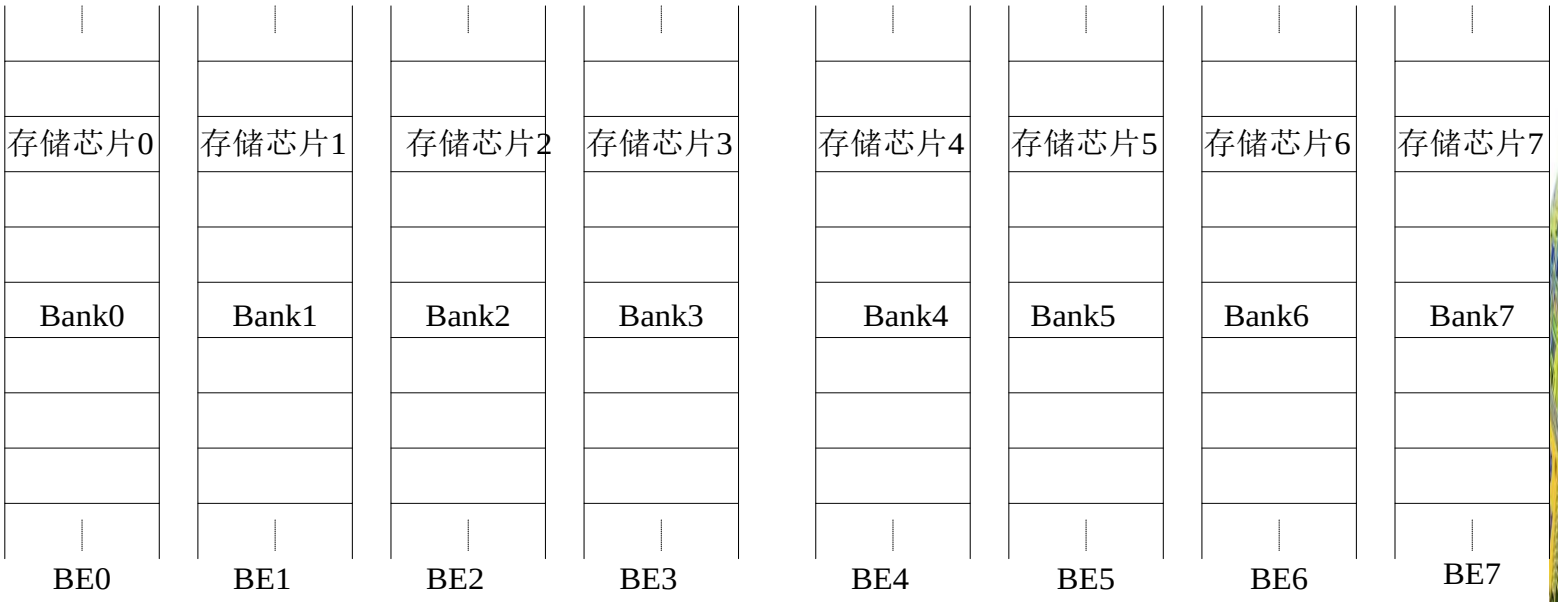
M=N=1000
CPU f=100MHz

4.2 内部存储器

- 存储器分块组织
  - 大部分微处理器支持访问8位，16位，32位到64位不等位宽的数据



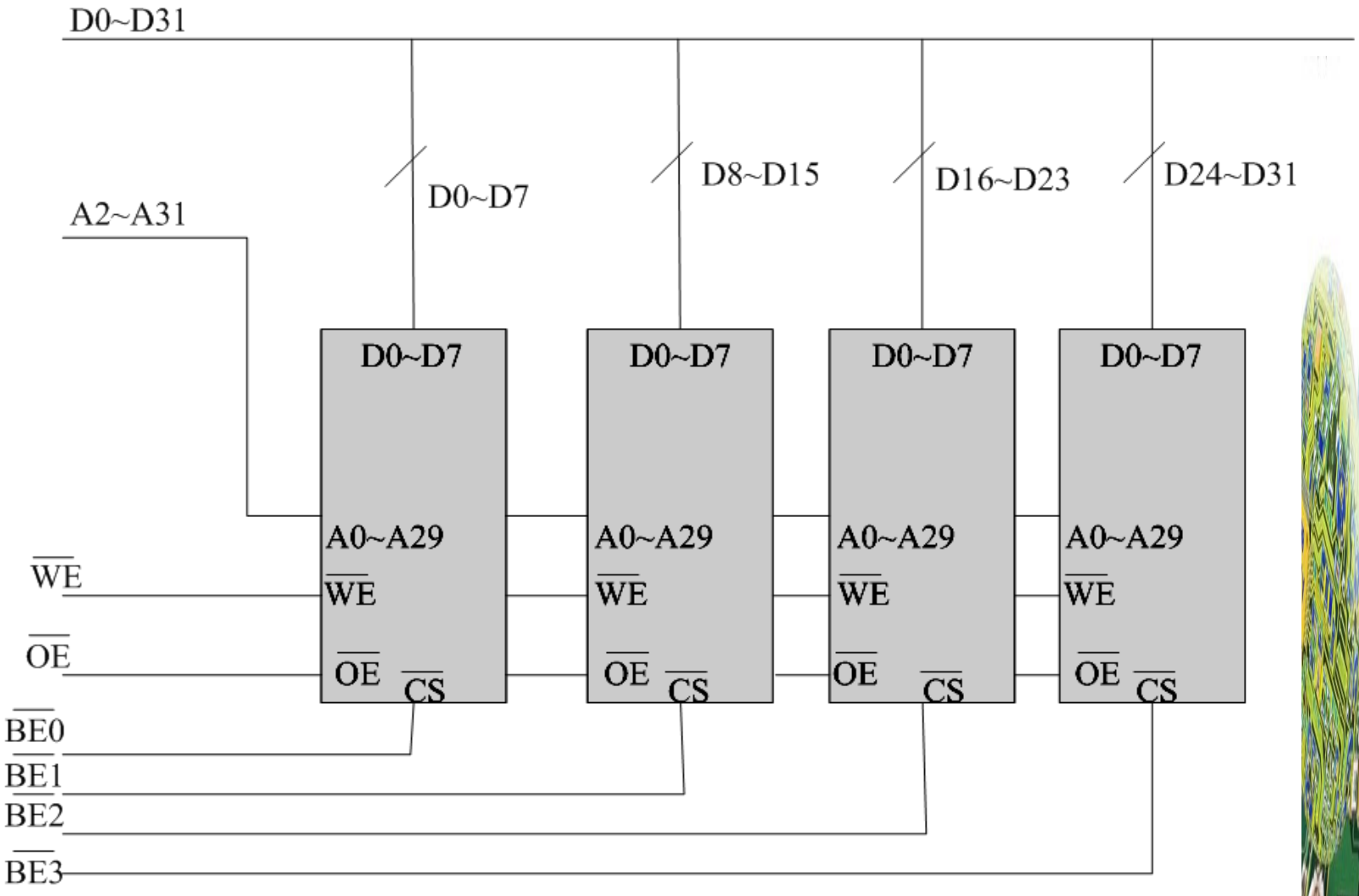
存储器分块示意图


存储芯片0
存储芯片1
存储芯片2
存储芯片3
存储芯片4
存储芯片5
存储芯片6
存储芯片7
Bank0
Bank1
Bank2
Bank3
Bank4
Bank5
Bank6
Bank7
BE0
BE1
BE2
BE3
BE4
BE5
BE6
BE7

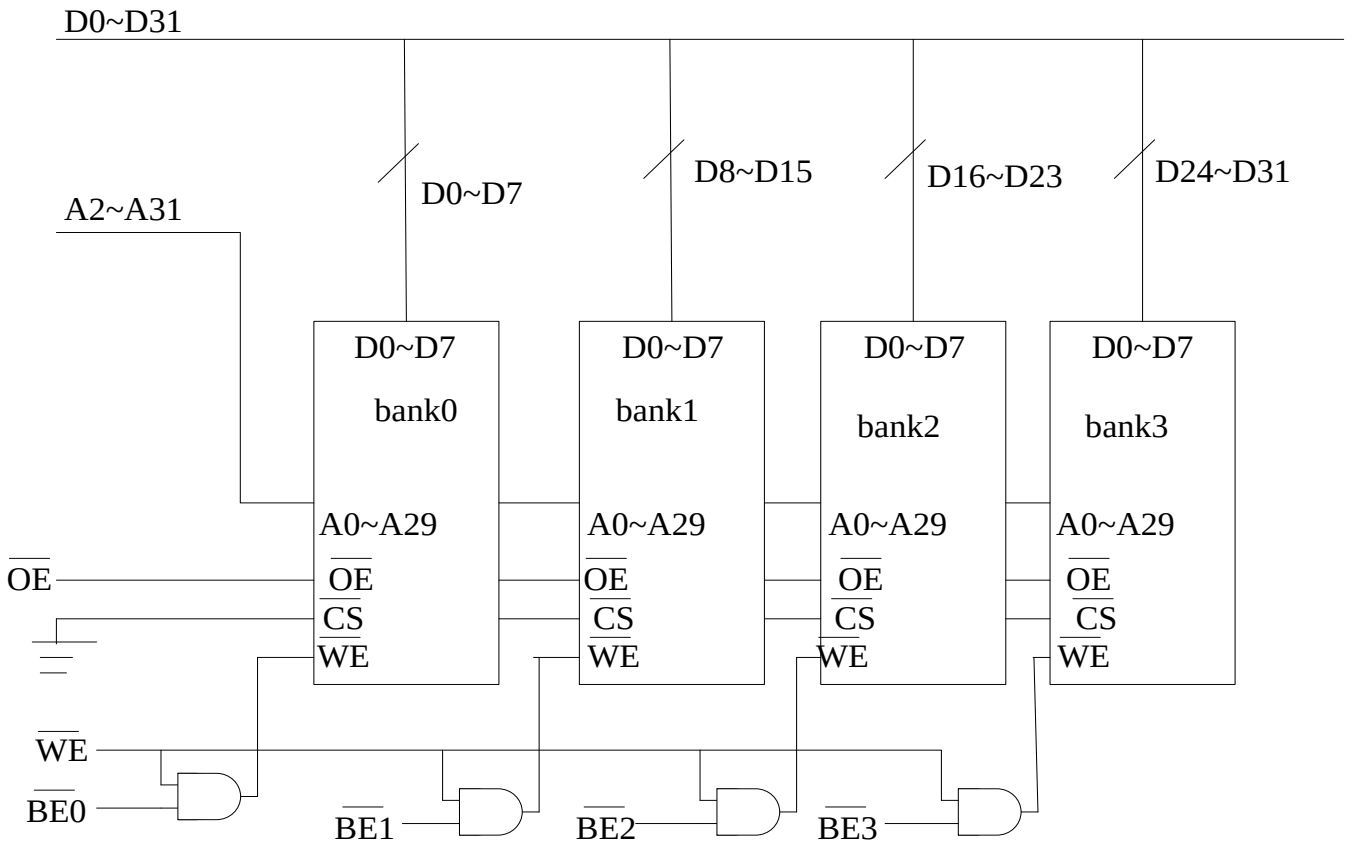
- 实现多类型数据访问的存储器接口控制有两种方式
  - 使用字节使能信号与高位地址译码产生存储芯片片选信号。
  - 使用字节使能信号与存储器写控制信号译码产生存储芯片写控制信号。

- 例4.1为一个32位的微处理器设计一个存储空间为4GB的SRAM存储器，要求支持字节、半字、字类型数据访问，采用1G*8bit的SRAM存储芯片，如何设计该存储器？
  - 存储器分为4块，表示存储器的A0~A1无效，存储芯片的A0~A29对应连接到存储器的A2~A31上

采用BE0~BE3分别控制4片芯片的片选线



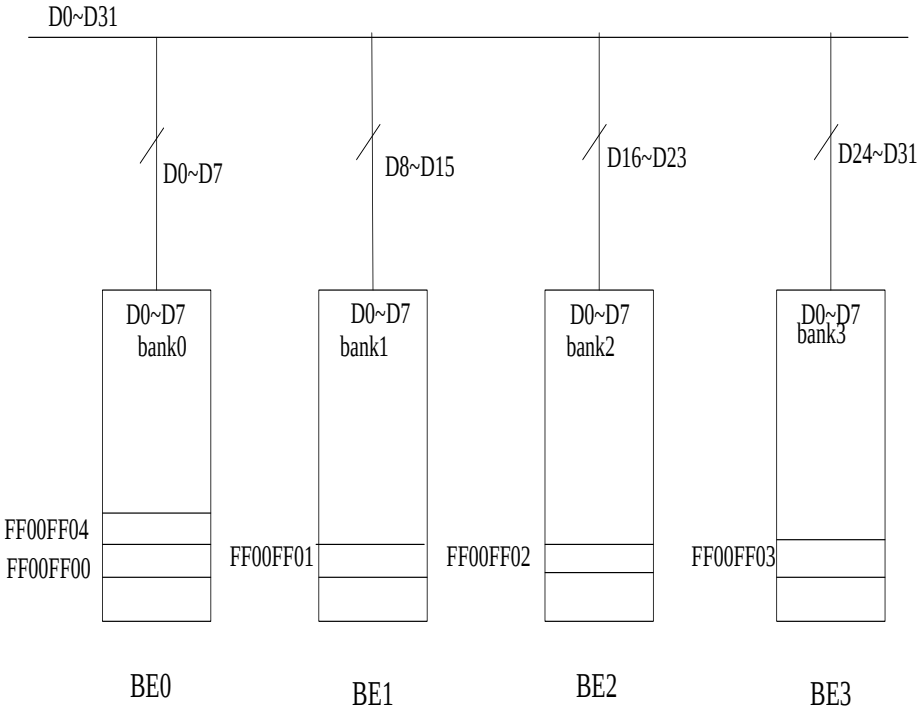
BE信号与存储器写信号控制存储芯片写信号实现不同类型的数据访问

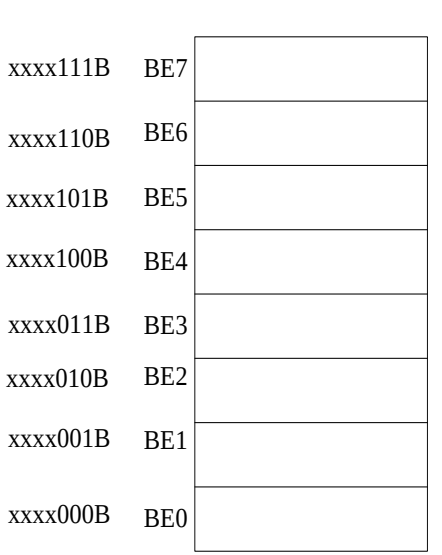


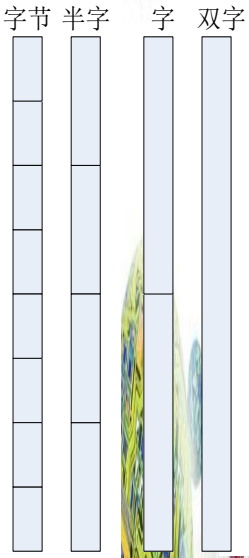
内存条

- DIMM内存条由8片8位数据宽度的同型号IC芯片组成，有的则由9片组成，增加的1片作校验位用。

内存访问边界对齐



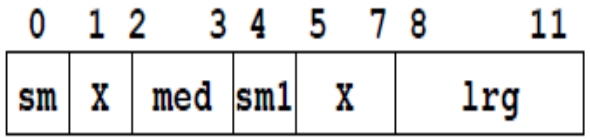




- 一次总线操作实现半字类型的数据访问要求字节使能信号从偶字节使能信号开始的连续两个字节有效，
- 一次总线操作实现字类型的数据访问要求字节使能信号从4字节使能信号开始的连续4个字节有效，
- 一次总线操作实现双字类型的数据访问要求字节使能信号从8字节使能信号开始的连续8个字节有效

- 例4.2 已知某32位计算机系统，编译器采用边界对其方式为数据分配内存空间，若定义了以下数据结构：

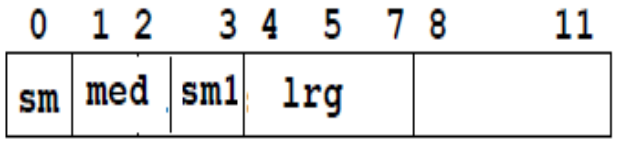
```
struct foo {
char sm; /*1字节*/
short med; /*2 字节*/
char sm1; /*1字节*/
int lrg; /*4字节*/
}
```



foo边界对齐的内存映像

浪费内存空间

```
struct foo1 {
char sm; /*1字节*/
char sm1; /*1字节*/
short med; /*2 字节*/
int lrg; /*4字节*/
}
```

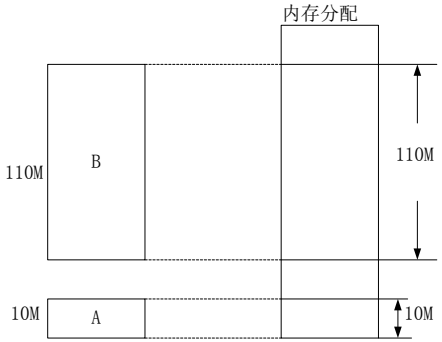


foo非边界对其的内存映像

增大访问时延

4.3内存管理

- 进程地址空间不隔离（误操作或恶意代码）
- 内存使用效率低（切换粒度为程序）
- 程序运行的地址不确定（软件设计复杂）



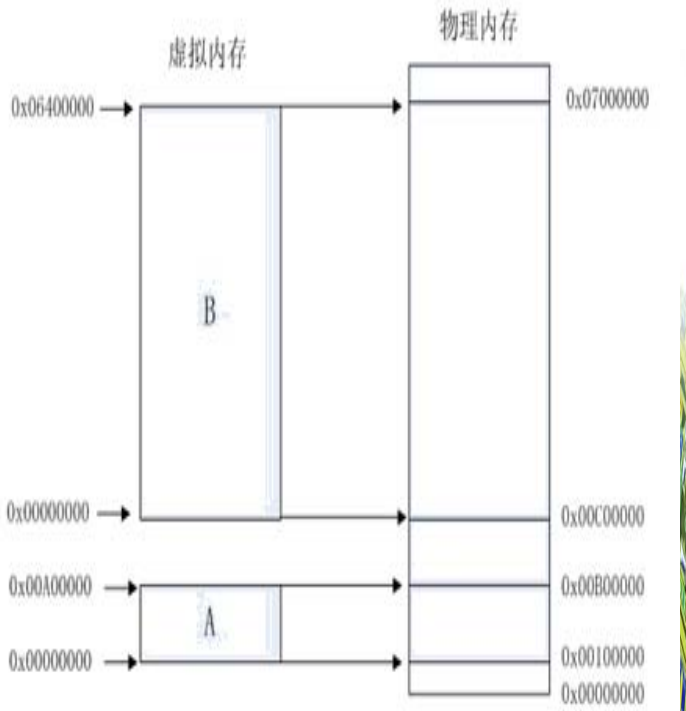
增加一个中间层，利用一种间接地址访问方法访问物理内存
程序中访问的内存地址不再是实际物理内存地址，而是一个虚拟地址，然后由操作系统将这个虚拟地址映射到适当的物理内存地址上

内存管理策略

- 分段：解决地址不确定以及冲突
- 分页：解决粗粒度调度
- 段页式

分段

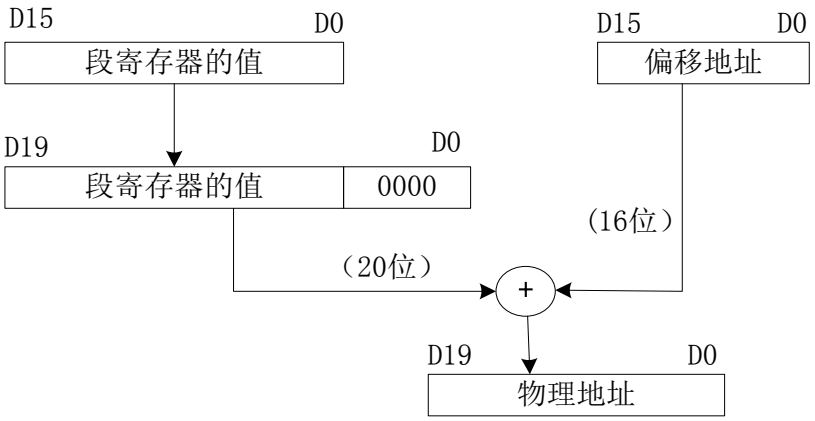
- 在虚拟地址空间和物理地址空间之间做一一映射。
- 比如说虚拟地址空间中某个10M大小的空间映射到物理地址空间中某个10M大小的空间
- 应用程序并不关心进程A究竟被映射到物理内存的哪块区域上，只需利用偏移地址访问内存单元即可



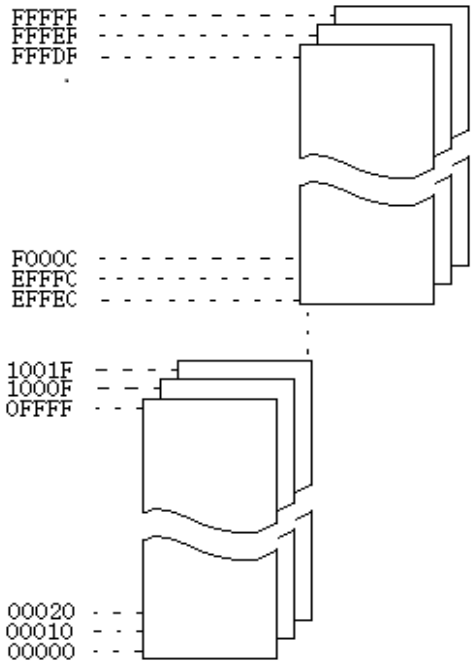
当微处理器要访问程序中的某个内存单元时，
将利用该程序映射到的物理地址空间段的起始地址（段地址）和偏移地址相结合的方式来实现寻址

分段实例

- Intel 实地址模式
- 保护模式



物理地址的形成



实地址模式分段

保护模式

字节7	段地址字节3	G	D	0	AV	界限高4位	字节6
字节5	访问权限	段地址字节2					字节4
字节3	段地址字节1	段地址字节0					字节2
字节1	界限字节1	界限字节0					字节0

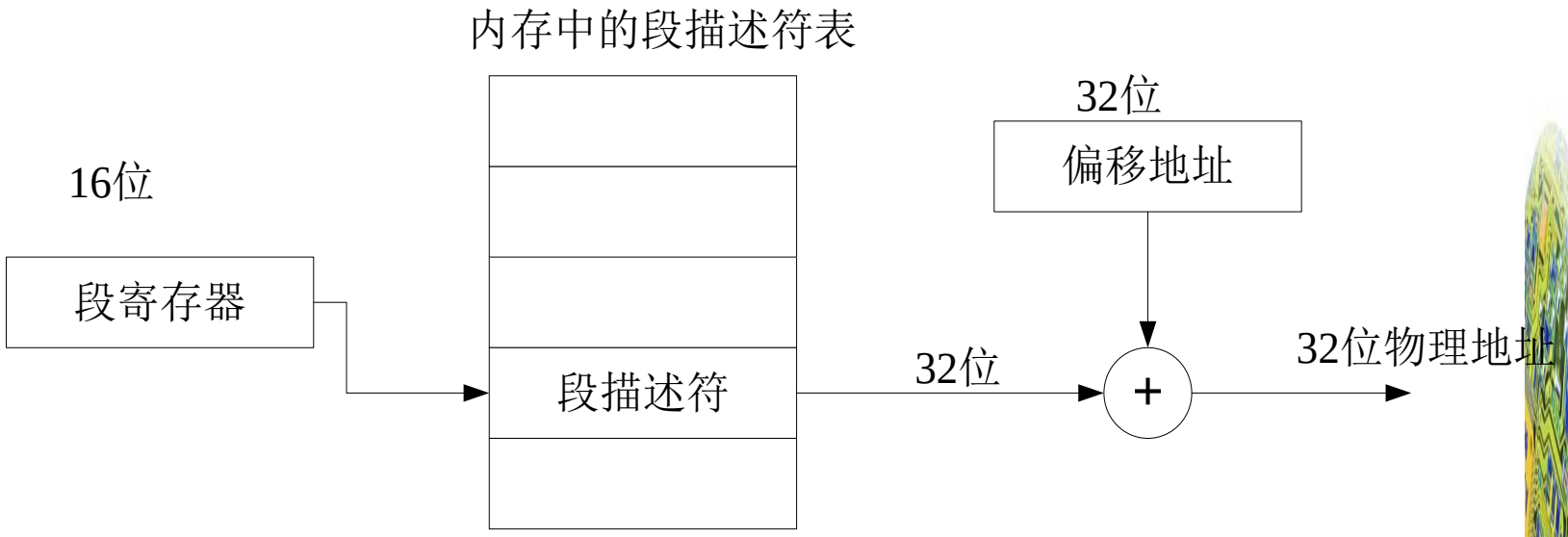
段描述符结构

G为界限的粒度，
当G=1时，段的最大偏移地址需要在20位界限表示的地址基础上乘以4K；
当G=0，段的最大偏移地址即为20位界限表示的地址

AV表示此段是否有效，AV=1，表示有效，AV=0，表示无效。
D表示指令模式，D=1，表示32位指令模式，D=0，表示16位指令模式。

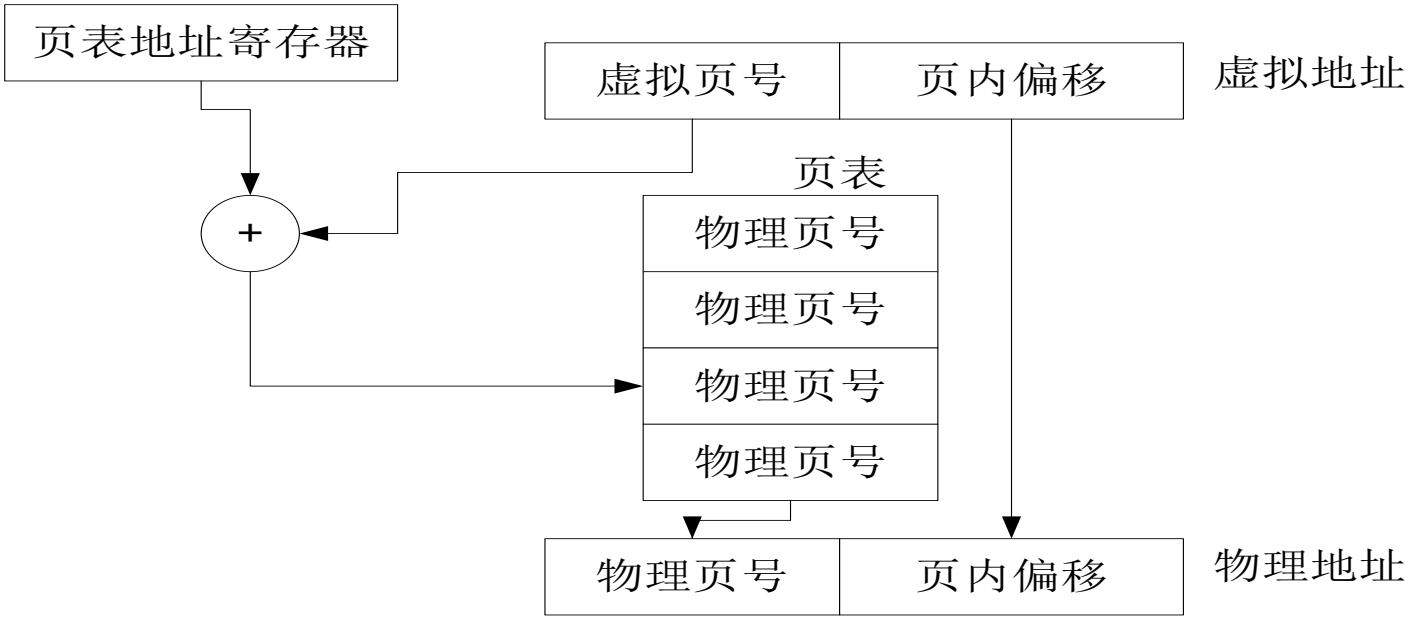
- 已知某32位intel微处理器的段描述符为0x34 d3 00 23 12 89 01 03，试指出该段描述符对应的段的起始地址与结束地址。
  - 段的起始地址：字节7，字节4，字节3，字节2，为：0x34231289.
  - 该段的界限为：字节6的低4位，字节1，字节0构成，即为：0x30103.
  - 该描述符的粒度G为1，因此实际的段界限为0x30103*4k,即为0x30103000.
  - 段的结束地址=起始地址+段界限-1.
  - 因此段的结束地址为：0x34231289+0x30103000-1=0x64334288.

保护模式下物理地址形成过程

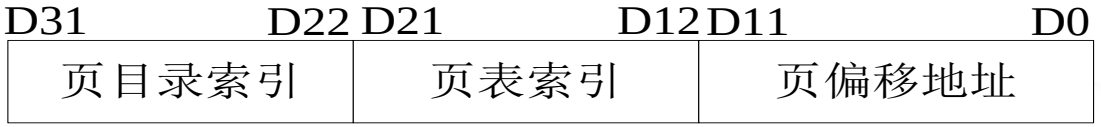


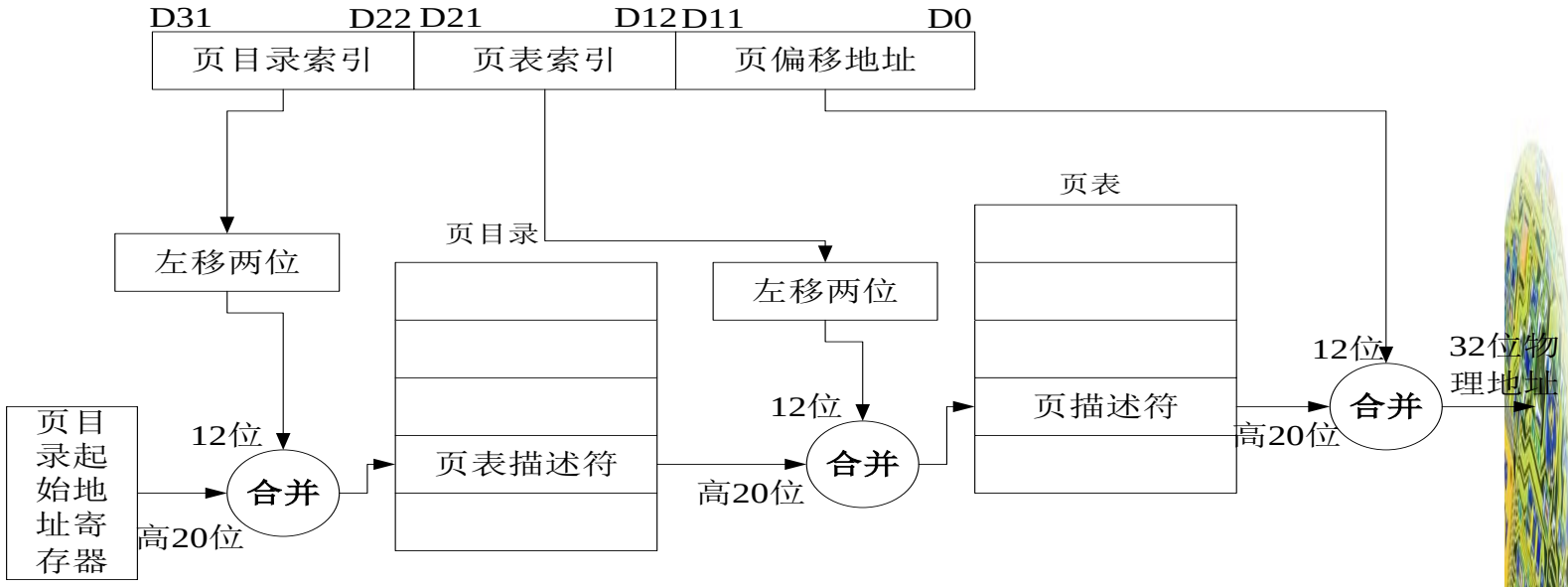
分页管理

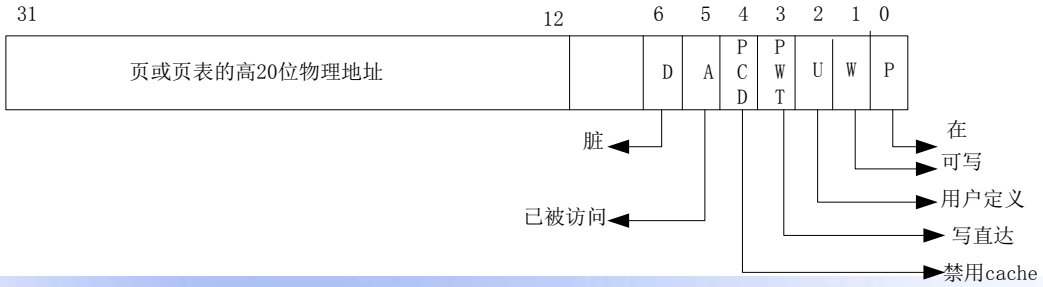
- 将地址空间分成许多相同大小的页
- 页的大小，决定了地址中用于寻址页内存储单元的位数



Intel 分页



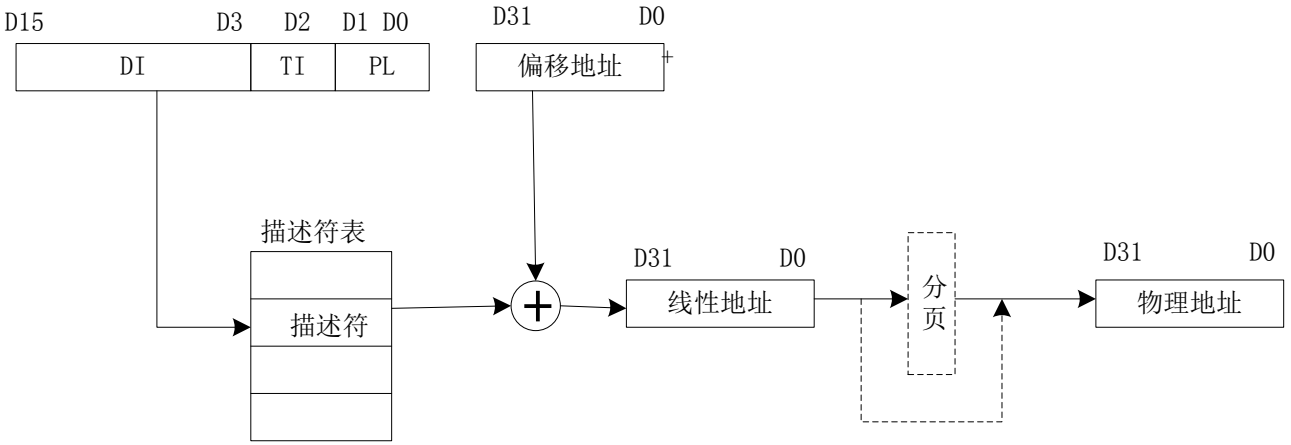




描述符结构

段页式管理

- 段页式管理是分段管理和分页管理的结合。
- 物理内存被等分成相同大小的页。
- 每个程序先按逻辑结构分段，每段再按照物理内存页大小分页。
- 程序访问内存时给出的逻辑地址需要先经过分段管理部件转换为线性地址，然后再经过分页管理部件转换为物理地址。



作业

- 2，5，6