

本节内容

基本分段存储管理方式

王道考研/CSKAOYAN.COM

1

知识总览

基本分段存储管理

与“分页”最大的区别就是——离散分配时所分配地址空间的基本单位不同

什么是分段（类似于分页管理中的“分页”）

什么是段表（类似于分页管理中的“页表”）

如何实现地址变换

分段、分页管理的对比

王道考研/CSKAOYAN.COM

2

分段

进程的地址空间：按照程序自身的逻辑关系划分为若干个段，每个段都有一个段名（在低级语言中，程序员使用段名来编程），每段从0开始编址

内存分配规则：以段为单位进行分配，每个段在内存中占据连续空间，但各段之间可以不相邻。

16KB

进程A

段名: MAIN
main 函数

段名: X
某个子函数

段名: D
保存全局变量

0号段 (7KB)
0 ~ 7K-1

1号段 (3KB)
0 ~ 3K-1

2号段 (6KB)
0 ~ 6K-1

0号段 (7KB)
40K ~ 80K

2号段 (6KB)
80K ~ 120K

1号段 (3KB)
120K ~ 126K

编译程序会将段名转换为段号

由于是按逻辑功能模块划分，用户编程更方便，程序的可读性更高

```
LOAD 1, [D] | <A>; //将分段D中A单元内的值读入寄存器1
STORE 1, [X] | <B>; //将寄存器1的内容存入X分段的B单元中
```

王道考研/CSKAOYAN.COM

3

分段

分段系统的逻辑地址结构由段号（段名）和段内地址（段内偏移量）所组成。如：

31	16	15	0
段号			段内地址		

段号的位数决定了每个进程最多可以分几个段
段内地址位数决定了每个段的最大长度是多少

在上述例子中，若系统是按字节寻址的，则
段号占16位，因此在该系统中，每个进程最多有 $2^{16} = 64K$ 个段
段内地址占16位，因此每个段的最大长度是 $2^{16} = 64KB$ 。

LOAD 1, [D] | <A>; //将分段D中A单元内的值读入寄存器1

STORE 1, [X] | ; //将寄存器1的内容存入X分段的B单元中

段名: MAIN
→ 段号: 0
main 函数

段名: X
→ 段号: 1
某个子函数

段名: D
→ 段号: 2
保存全局变量

单元

<A>单元

写程序时使用的段名 [D]、[X] 会被编译程序翻译成对应段号

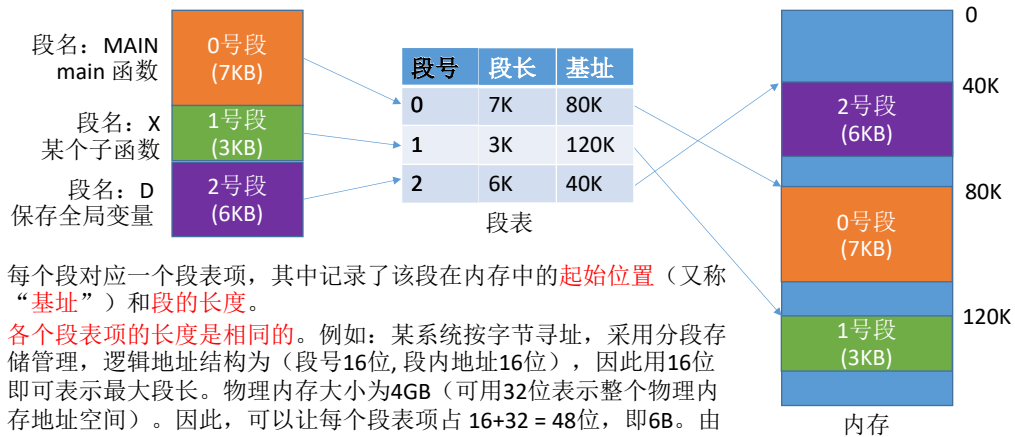
<A>单元、单元会被编译程序翻译成段内地址

王道考研/CSKAOYAN.COM

4

段表

问题：程序分多个段，各段离散地装入内存，为了保证程序能正常运行，就必须能从物理内存中找到各个逻辑段的存放位置。为此，需为每个进程建立一张段映射表，简称“**段表**”。

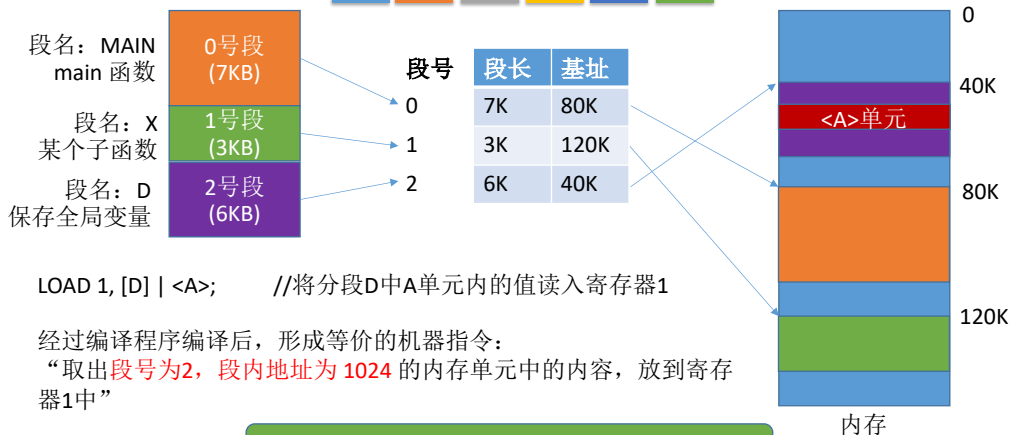


1. 每个段对应一个段表项，其中记录了该段在内存中的**起始位置**（又称“**基址**”）和**段的长度**。
2. **各个段表项的长度是相同的**。例如：某系统按字节寻址，采用分段存储管理，逻辑地址结构为（段号16位，段内地址16位），因此用16位即可表示最大段长。物理内存大小为4GB（可用32位表示整个物理内存地址空间）。因此，可以让每个段表项占 $16+32=48$ 位，即6B。由于段表项长度相同，因此**段号可以是隐含的，不占存储空间**。若段表存放的起始地址为M，则K号段对应的段表项存放的地址为 $M + K*6$

王道考研/CSKAOYAN.COM

5

地址变换



LOAD 1, [D] | <A>; //将分段D中A单元内的值读入寄存器1

经过编译程序编译后，形成等价的机器指令：

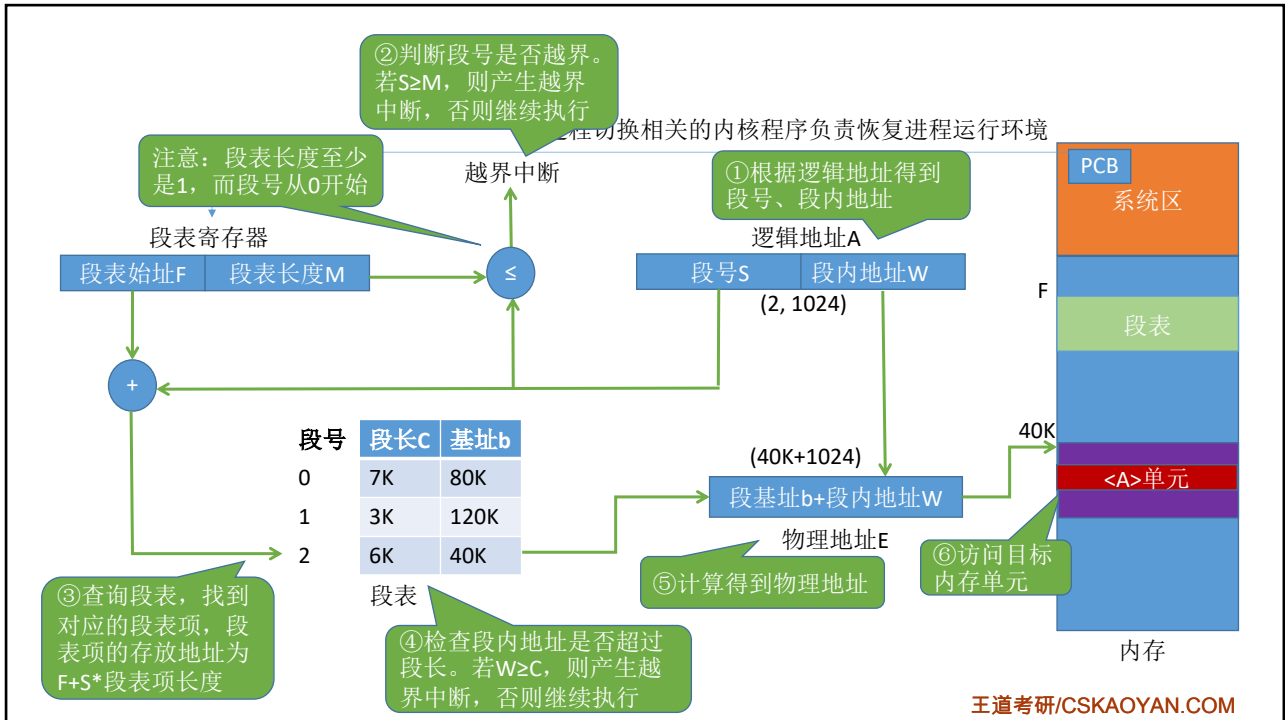
“取出**段号为2**，**段内地址为1024**的内存单元中的内容，放到寄存器1中”

CPU执行指令时需要将逻辑地址变换为物理地址

机器指令中的逻辑地址用二进制表示：**0000000000000010**0000000100000000

王道考研/CSKAOYAN.COM

6



7

分段、分页管理的对比

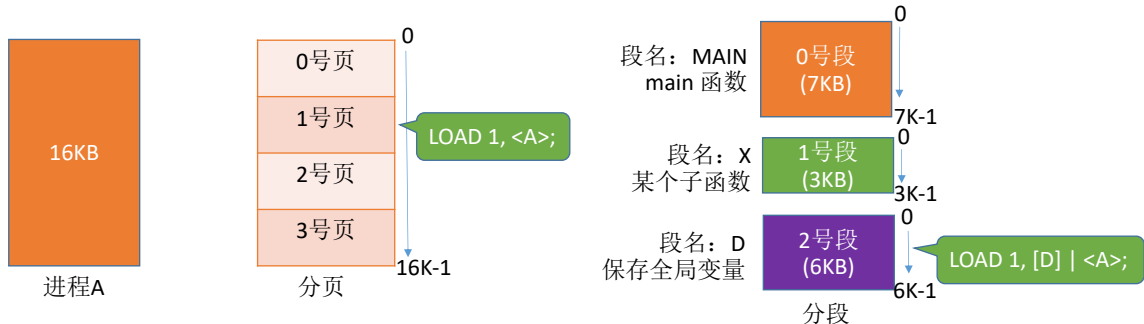
页是信息的物理单位。分页的主要目的是为了实现离散分配，提高内存利用率。分页仅仅是系统管理上的需要，完全是系统行为，对用户是不可见的。

段是信息的逻辑单位。分段的主要目的是更好地满足用户需求。一个段通常包含着一组属于一个逻辑模块的信息。分段对用户是可见的，用户编程时需要显式地给出段名。

页的大小固定且由系统决定。段的长度却不固定，决定于用户编写的程序。

分页的用户进程地址空间是一维的，程序员只需给出一个记忆符即可表示一个地址。

分段的用户进程地址空间是二维的，程序员在标识一个地址时，既要给出段名，也要给出段内地址。



8

分段、分页管理的对比

分段比分页更容易实现信息的共享和保护。

不能被修改的代码称为**纯代码**或**可重入代码**（不属于临界资源），这样的代码是可以共享的。可修改的代码是不能共享的（比如，有一个代码段中有很多变量，各进程并发地同时访问可能造成数据不一致）

该功能段用来判断缓冲区此时是否可访问。允许所有生产者、消费者进程共享访问

生产者进程

将生产者进程分段

只需让各进程的段表项指向同一个段即可实现共享

生产者进程A的段表

段号	段长	基址
0	7K	80K
1	3K	120K
2	6K	40K

消费者进程B的段表

段号	段长	基址
0	21K	200K
1	3K	120K

比如，有一个代码段只是简单的输出“Hello World!”

内存

王道考研/CSKAOYAN.COM

9

分段、分页管理的对比

分段比分页更容易实现信息的共享和保护。

将生产者进程分段

将生产者进程分页

页面不是按逻辑模块划分的。这就很难实现共享。

如果让消费者进程的某个页表项指向这个页面，显然不合理，因为这个页面中的橙色部分是不允许共享的，只有绿色部分可以

1、2号页面只有一部分允许其他进程访问，因此很难用页表实现信息保护

生产者进程A的段表

段号	段长	基址	是否允许其他进程访问
0	7K	80K	不允许
1	3K	120K	允许
2	6K	40K	不允许

生产者进程A的页表

页号	基址	是否允许其他进程访问
0	...	不允许
1	...	允许
2	...	允许
3	...	不允许

王道考研/CSKAOYAN.COM

10

分段、分页管理的对比

页是信息的物理单位。分页的主要目的是为了实现离散分配，提高内存利用率。分页仅仅是系统管理上的需要，完全是系统行为，对用户是不可见的。

段是信息的逻辑单位。分段的主要目的是更好地满足用户需求。一个段通常包含着一组属于一个逻辑模块的信息。分段对用户是可见的，用户编程时需要显式地给出段名。

页的大小固定且由系统决定。段的长度却不固定，决定于用户编写的程序。

分页的用户进程地址空间是一维的，程序员只需给出一个记忆符即可表示一个地址。

分段的用户进程地址空间是二维的，程序员在标识一个地址时，既要给出段名，也要给出段内地址。

分段比分页更容易实现信息的共享和保护。不能被修改的代码称为纯代码或可重入代码（不属于临界资源），这样的代码是可以共享的。可修改的代码是不能共享的。

访问一个逻辑地址需要几次访存？

分页（单级页表）：第一次访存——查内存中的页表，第二次访存——访问目标内存单元。总共两次访存

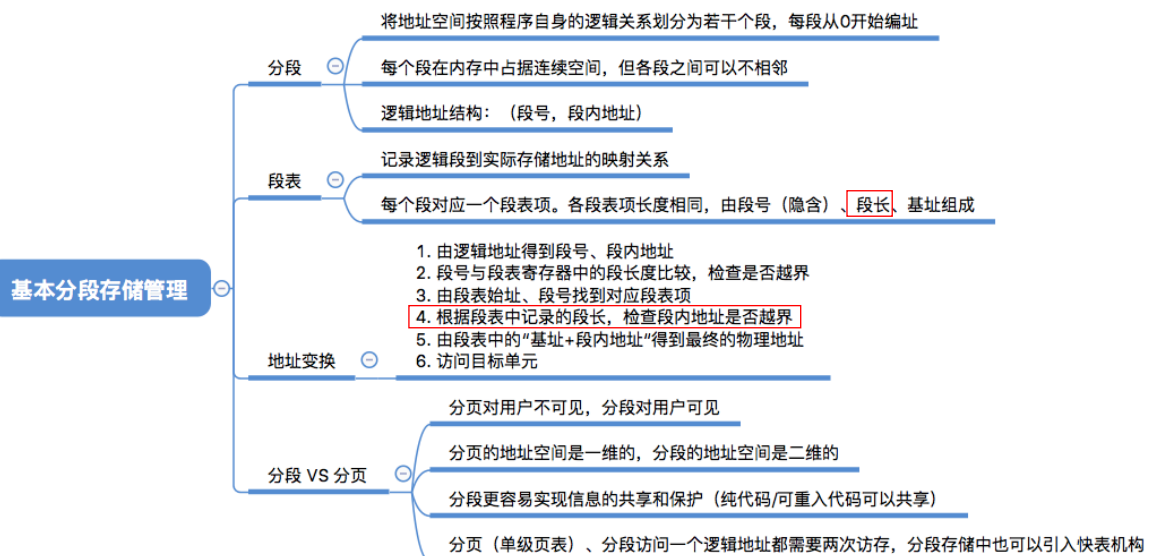
分段：第一次访存——查内存中的段表，第二次访存——访问目标内存单元。总共两次访存

与分页系统类似，分段系统中也可以引入快表机构，将近期访问过的段表项放到快表中，这样可以少一次访问，加快地址变换速度。

王道考研/CSKAOYAN.COM

11

知识回顾与重要考点



王道考研/CSKAOYAN.COM

12



@王道论坛



等撩

@王道计算机考研备考
@王道咸鱼老师-计算机考研
@王道楼楼老师-计算机考研



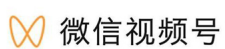
等撩



@王道计算机考研



@王道计算机考研



@王道计算机考研



@王道在线