操作系统

第五章 存储管理

文艳军（教授）
计算机学院

本章讲授计划

① 连续空间分配

② 段式管理

③ 页式管理

④ 段页式管理

⑤ 页式虚存管理

Linux 0.11

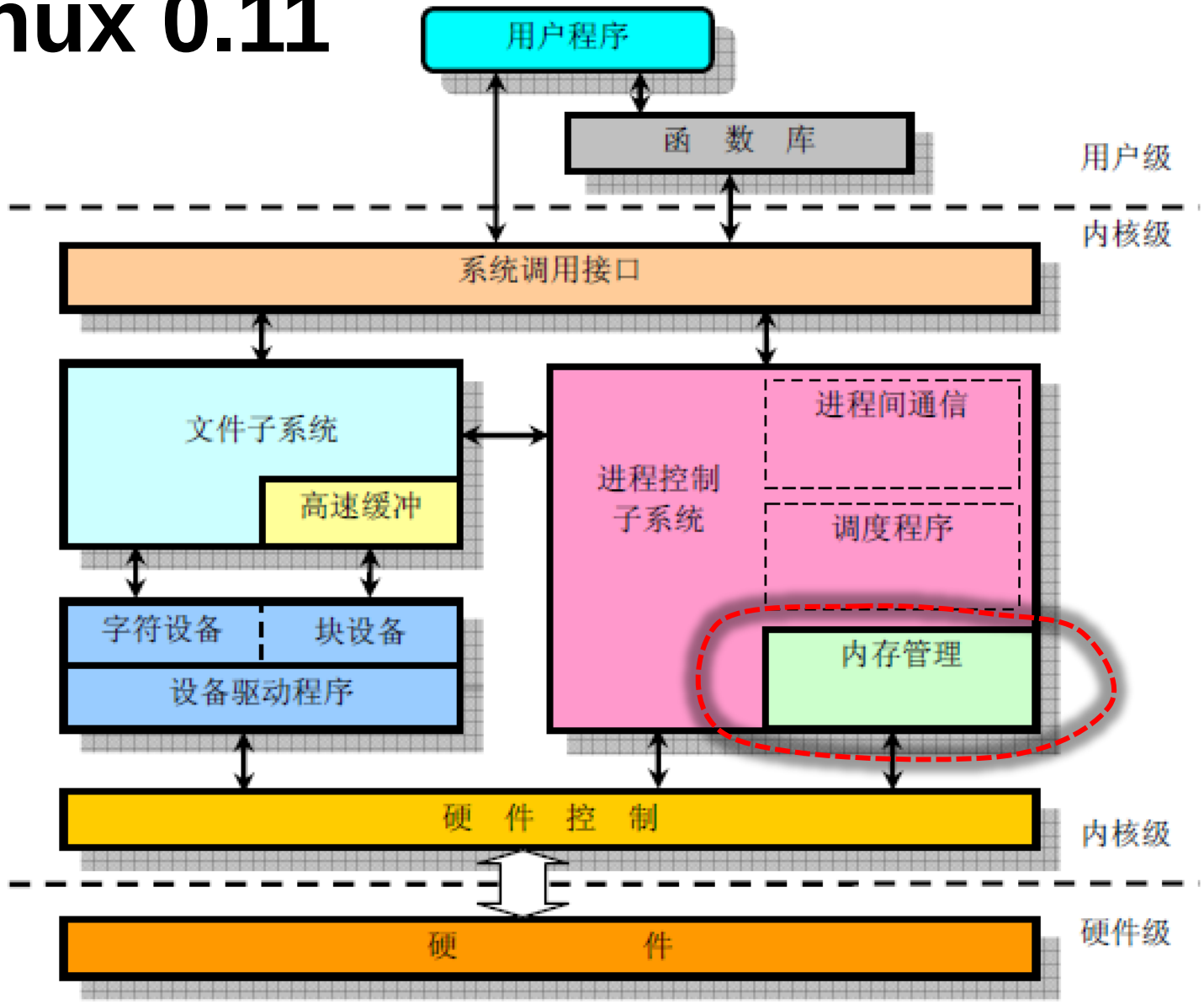


图 2-4 内核结构框图

第五章 存储管理

- 主题：如何将内存分配给进程使用。

目录

一. 连续空间分配

1. 单道连续分配
2. 多道固定分区法
3. 多道连续可变分区法

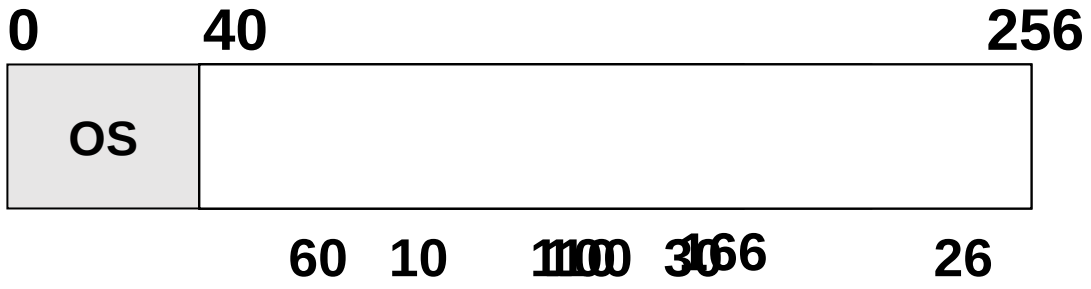
特征：每个进程占据连续的物理内存。

3. 多道连续可变分区法

思想： 系统设置一个空闲块队列，初始状态时队列中只有一个连续的空闲块。作业到达后，以某种策略分配空间。作业撤离时，将释放的空间加入空闲队列，相邻区合并。

举例：假设任一时间段内，内存中作业采用时间片轮转调度且运行时间片为1。

作业队列次序	所需存储量	运行时间	已运行时间
1	60	10	5 5
2	100	5	5
3	30	20	5 5 3 7
4	70	8	5 3
5	50	15	3 7



3. 多道连续可变分区法

- 只有外部碎片，没有内部碎片。
- **紧致**：通过移动作业位置可以将零散的空闲块连接成大块，减少外部碎片。要求作业动态可浮动。

  - **内部碎片**：内存某存储区间大于其存放作业空间的部分。
  - **外部碎片**：内存某存储区间容不下要运行的作业时。

小结

一. 连续空间分配

多道连续可变分区法

二. 不连续空间分配

段式管理、页式管理、段页式管理、

“进程创建”中的内存拷贝

三. 页式虚存管理

目录

一. 连续空间分配

多道连续可变分区法

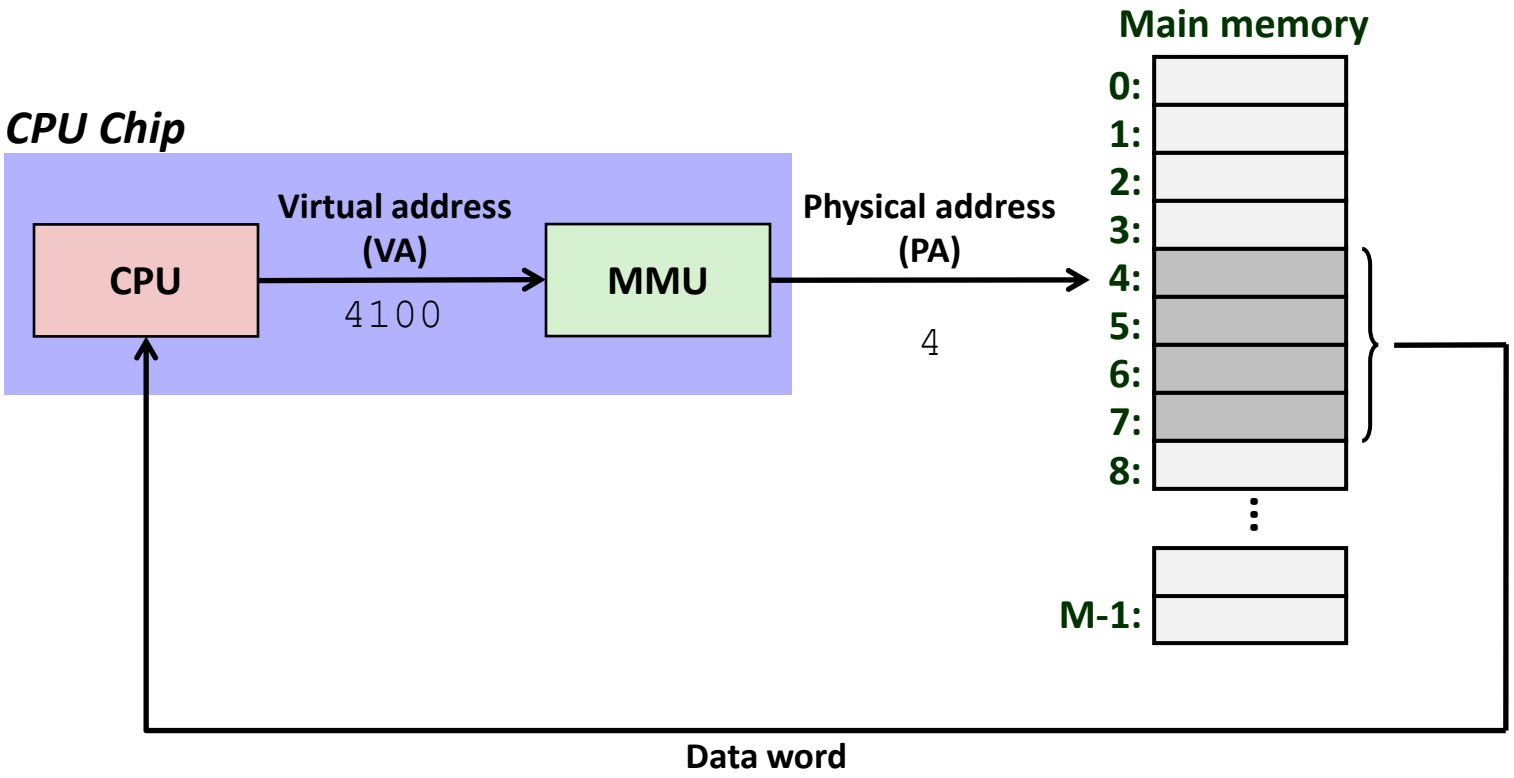
二. 不连续空间分配

段式管理、页式管理、段页式管理、

“进程创建”中的内存拷贝

三. 页式虚存管理

二. 不连续空间分配



- Used in all modern servers, desktops, and laptops
- One of the great ideas in computer science

二. 不连续空间分配

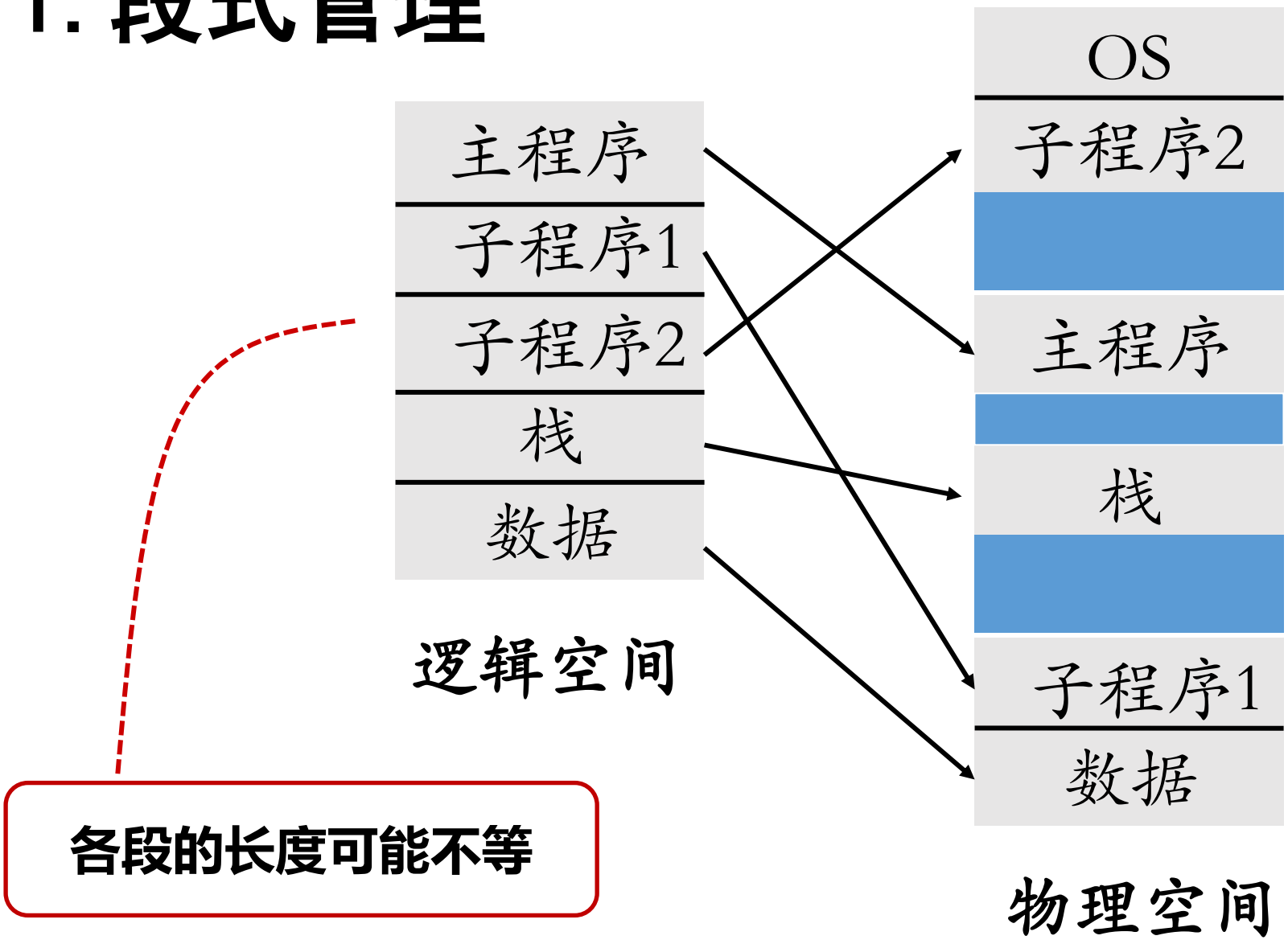
1. 段式管理
2. 页式管理
3. 段页式管理

特征： 每个进程可占据不连续的物理内存。

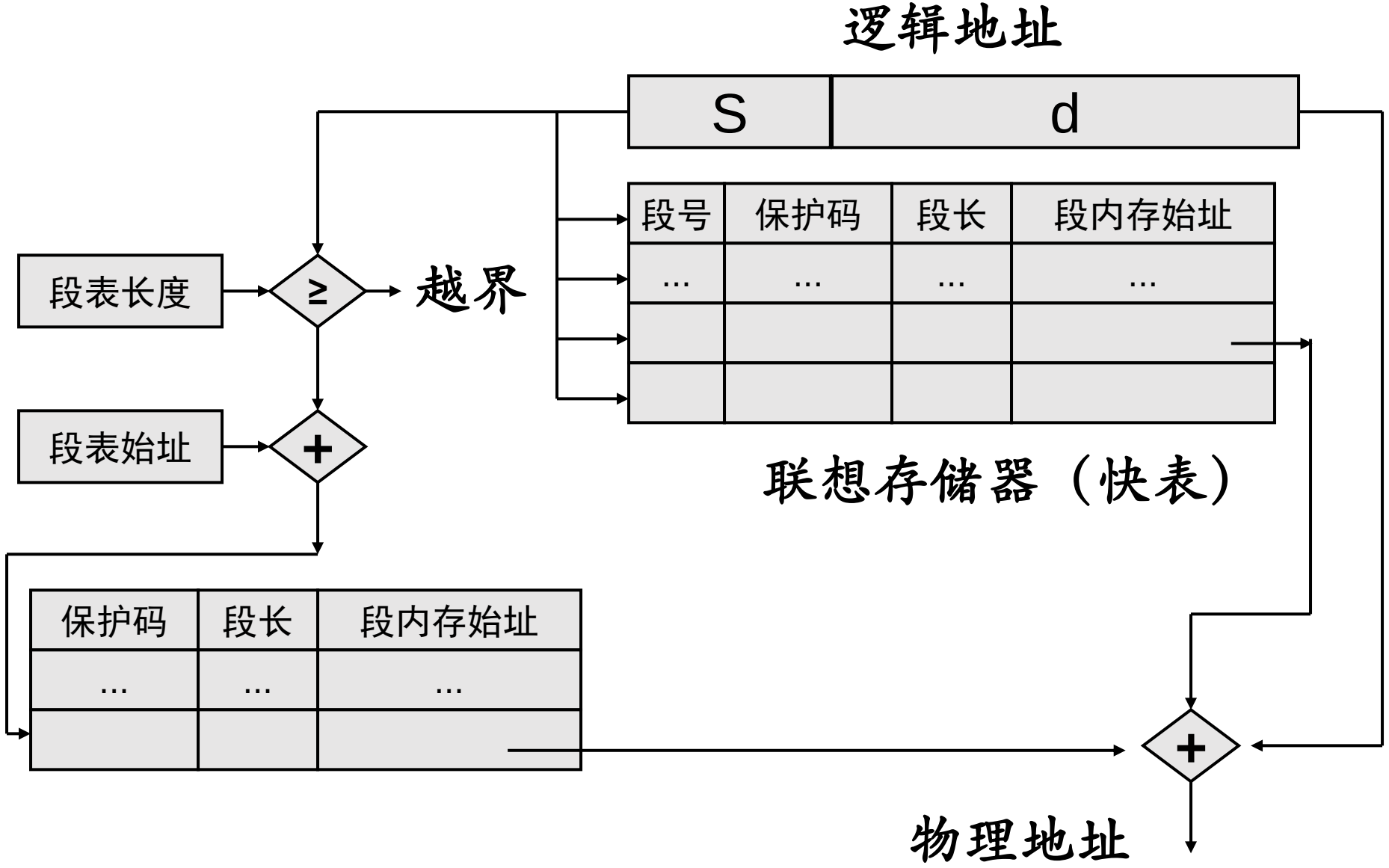
1. 段式管理

特点：以进程的自然段为单位分配物理内存，段内连续，段间可不连续。

1. 段式管理



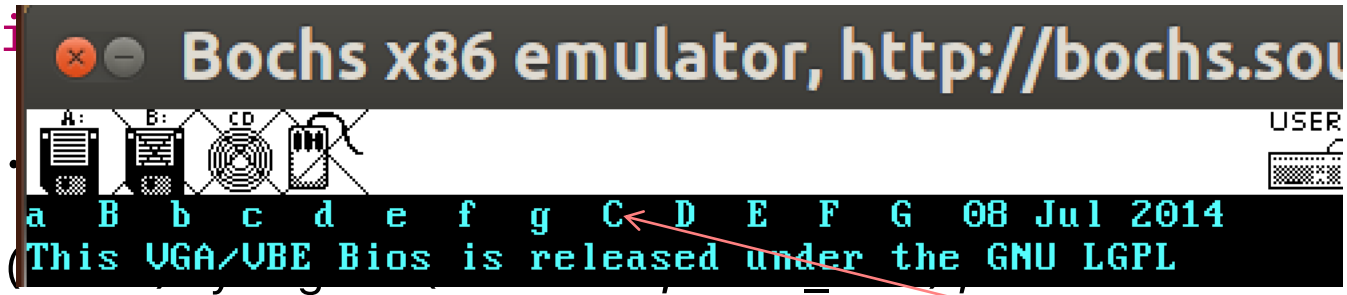
段式地址转换过程



1.3 版内核的初始化程序

```
static int mynext = 0;
void main(void) {
  ...
  if (!fork()) {
    (void)mysignal(SIGALRM, SIG_IGN);
    for(;;) {
      if (mynext < 7) {
        output_char('A' + mynext);
        mynext++;
      }
      alarm(1);
      pause();
    }
  }
  for(;;) {
    if (mynext < 7) {
      output_char('a' + mynext);
      mynext++;
    }
    pause();
  }
}
```

1.3 版内核的初始化程序




```
stati
void
   ...
   if
      (
      for(;;) {
        if (mynext < 7) {
          output_char('A' + mynext);
          mynext++;
        }
        alarm(1);
        pause();
      }
   }
   for(;;) {
     if (mynext < 7) {
       output_char('a' + mynext);
       mynext++;
     }
     pause();
   }
}
```

为什么不是H?

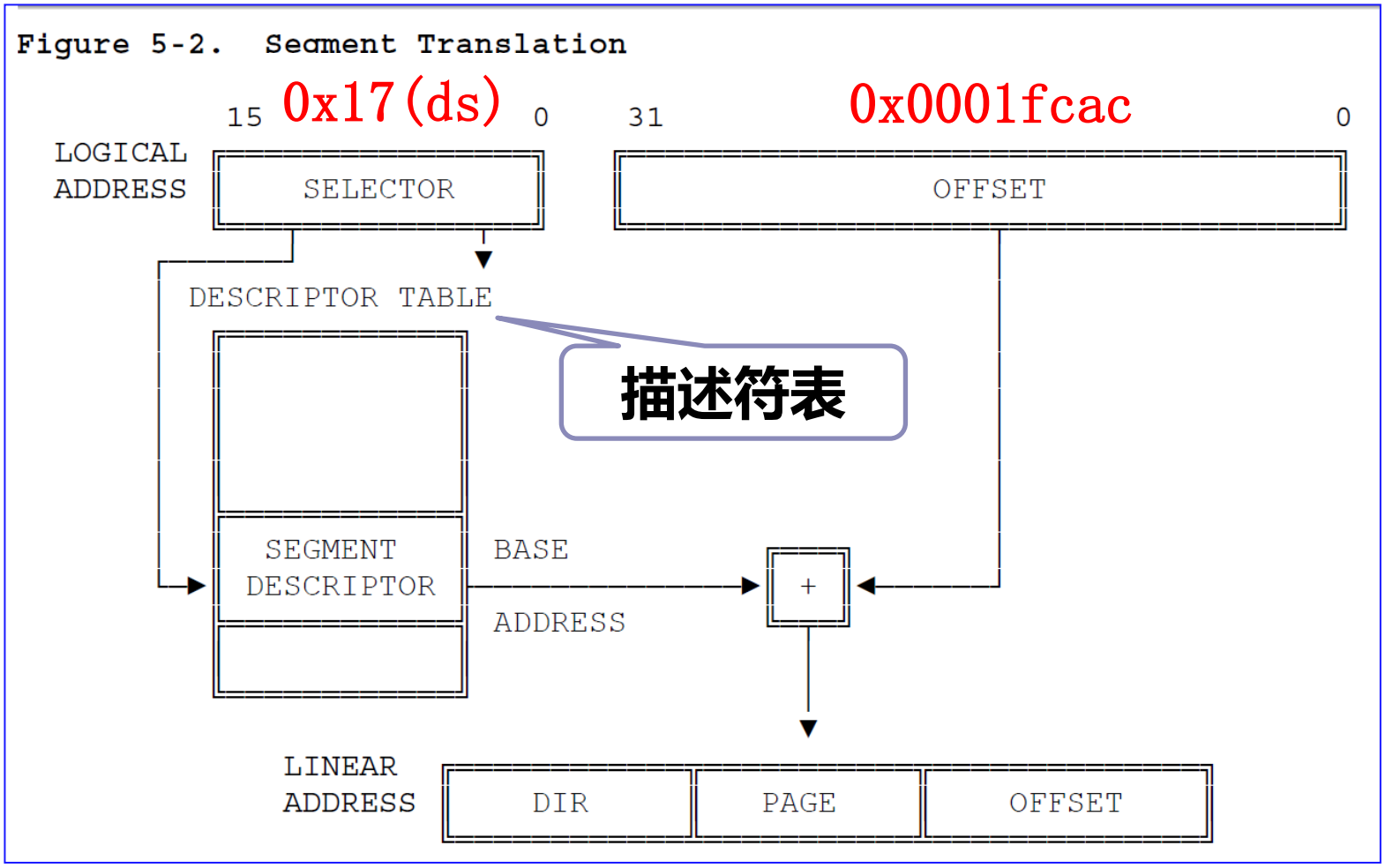
1. 段式管理

- 80x86 CPU的默认段寄存器
  - 代码：CS
  - 数据：DS
  - 栈：SS

```
154         if (mynext < 7) {
155           output_char('A' + mynext);
156           mynext++;
157         }
```

```
=> 0x699f <main+404>:   mov     0x1fcac,%eax
   0x69a4 <main+409>:   add     $0x41,%eax
   0x69a7 <main+412>:   movsbl %al,%eax
   0x69aa <main+415>:   mov     %eax,(%esp)
   0x69ad <main+418>:   call    0x7a17 <output_char>
```

实例:80x86 CPU的分段机制



地址0x17:0x0001fcac在内存的什么位置？

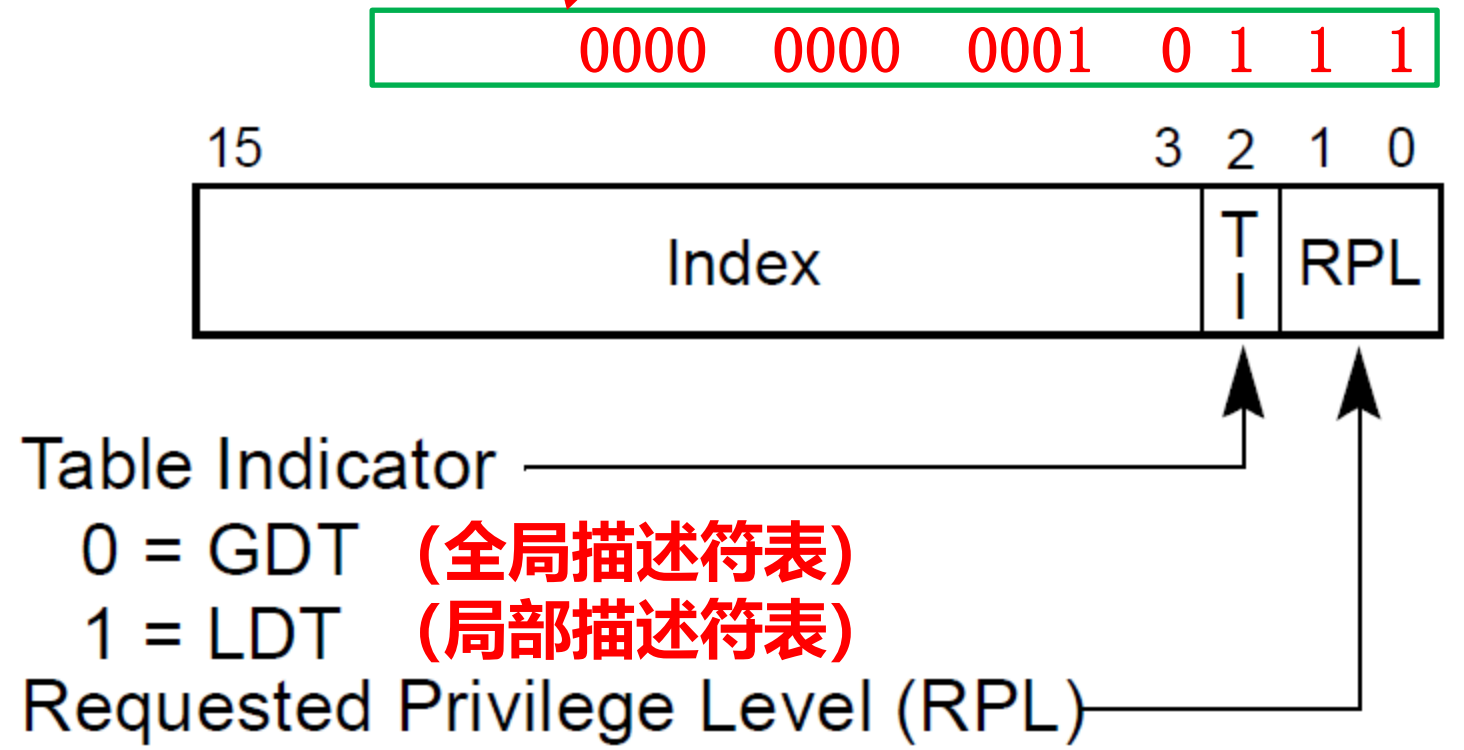
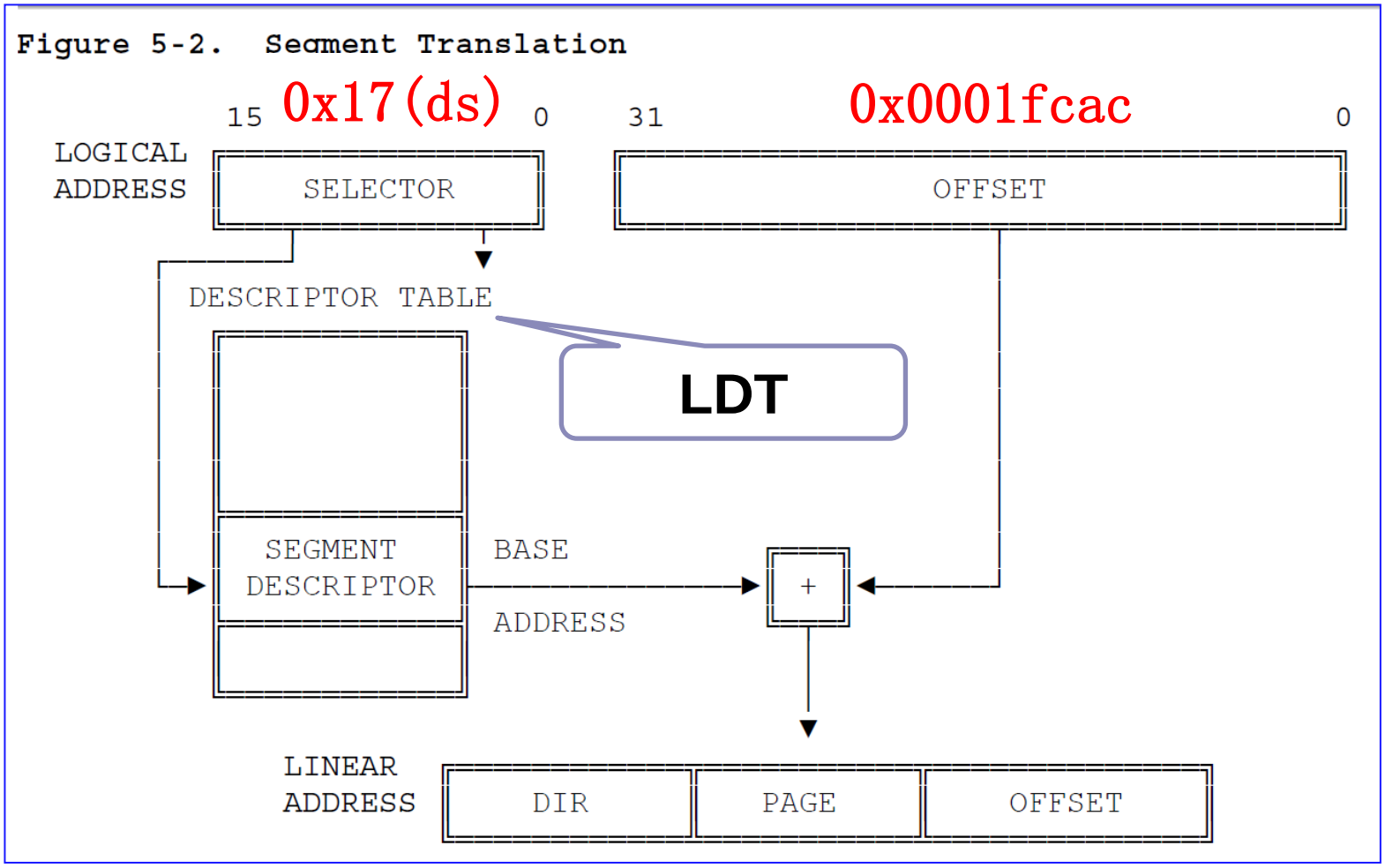
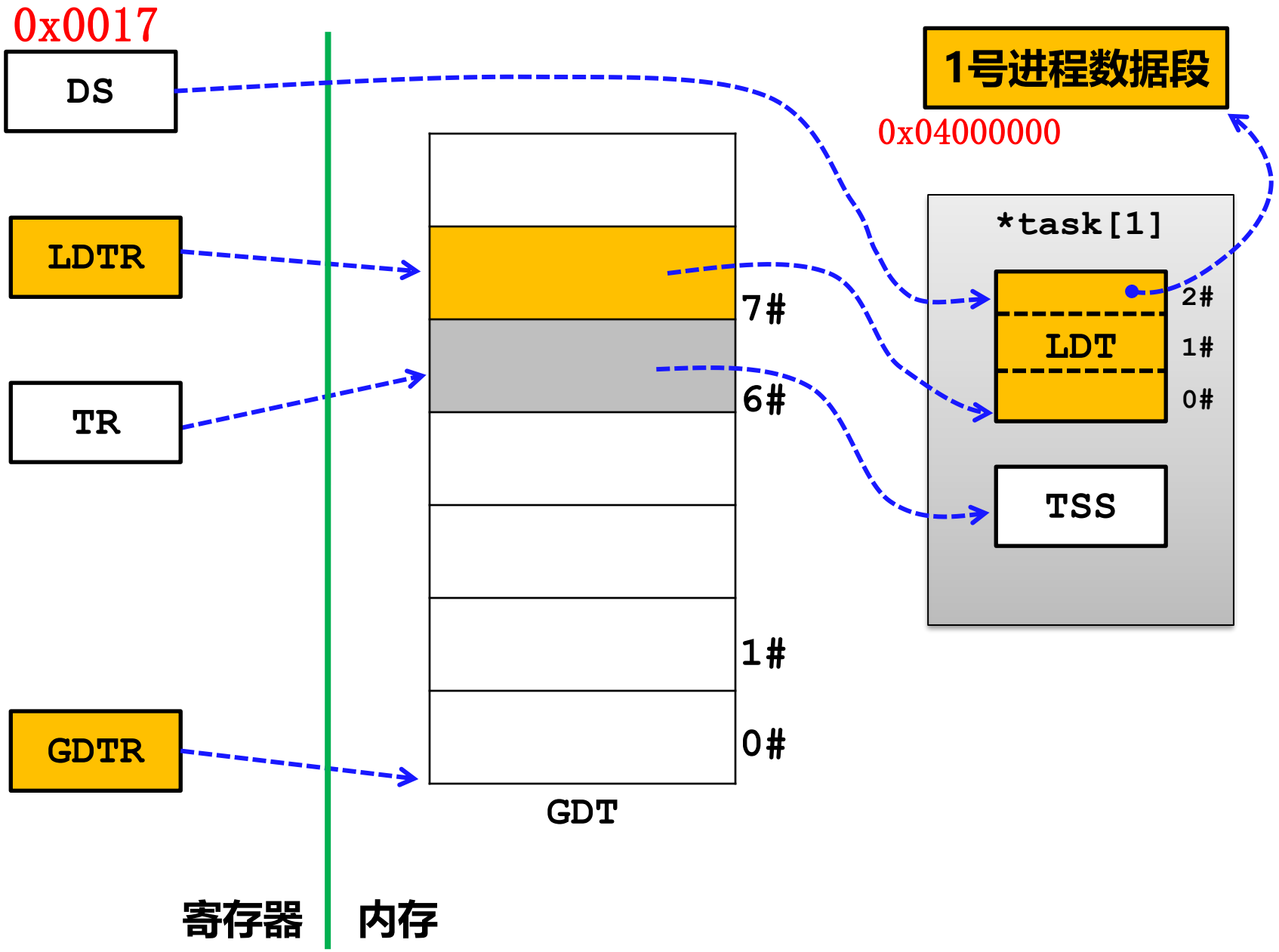


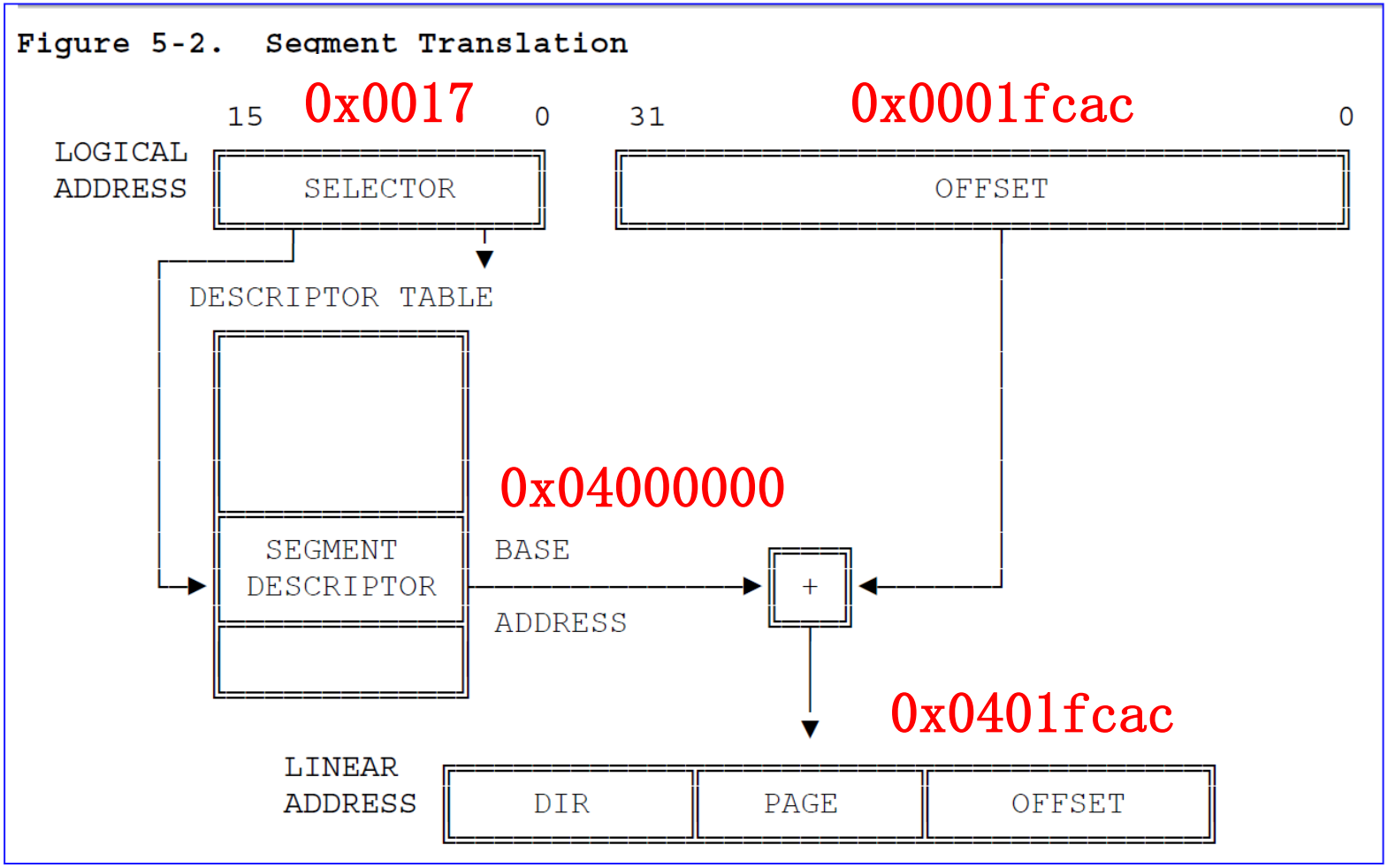
Figure 3-6. Segment Selector

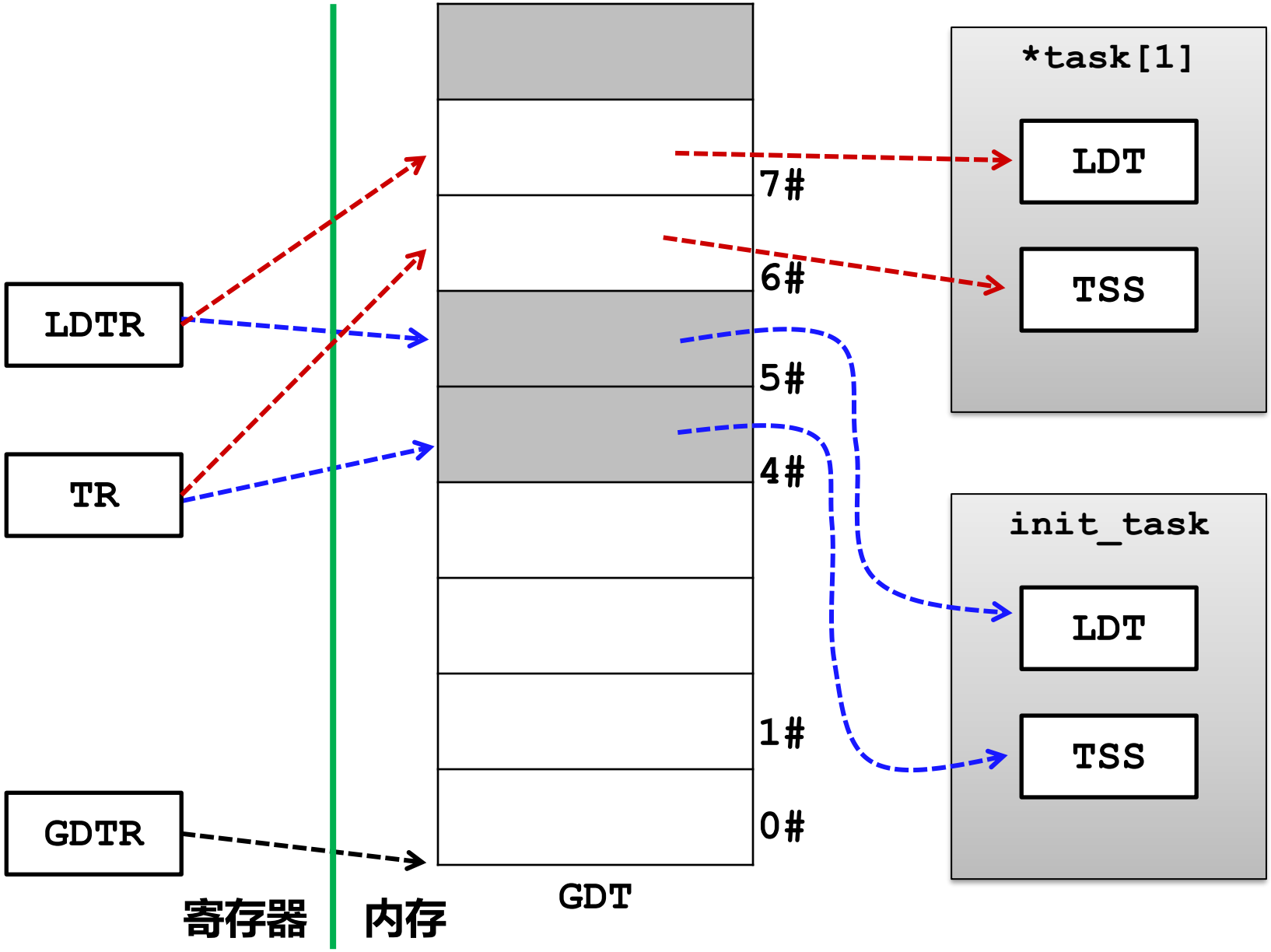
实例:80x86 CPU的分段机制

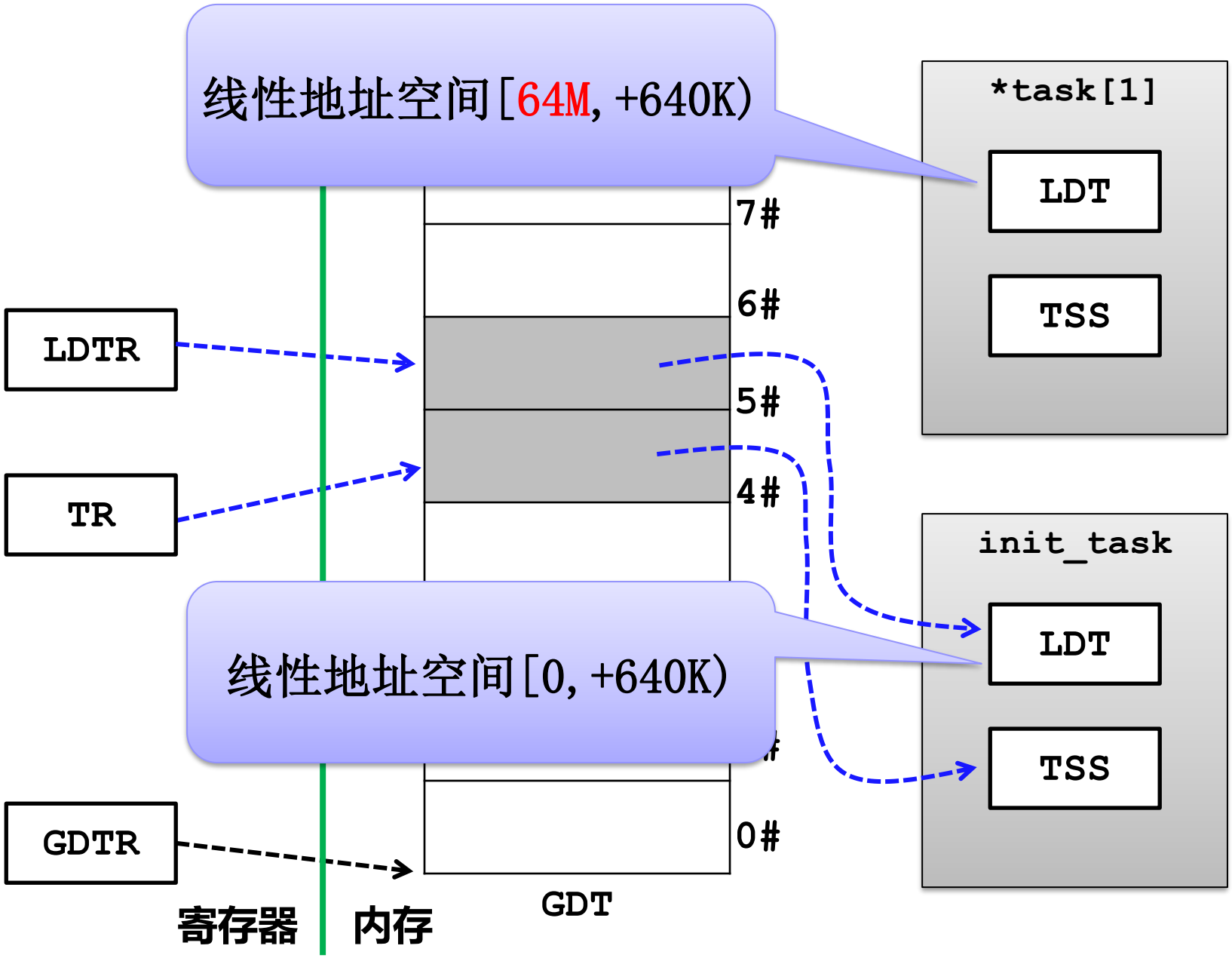


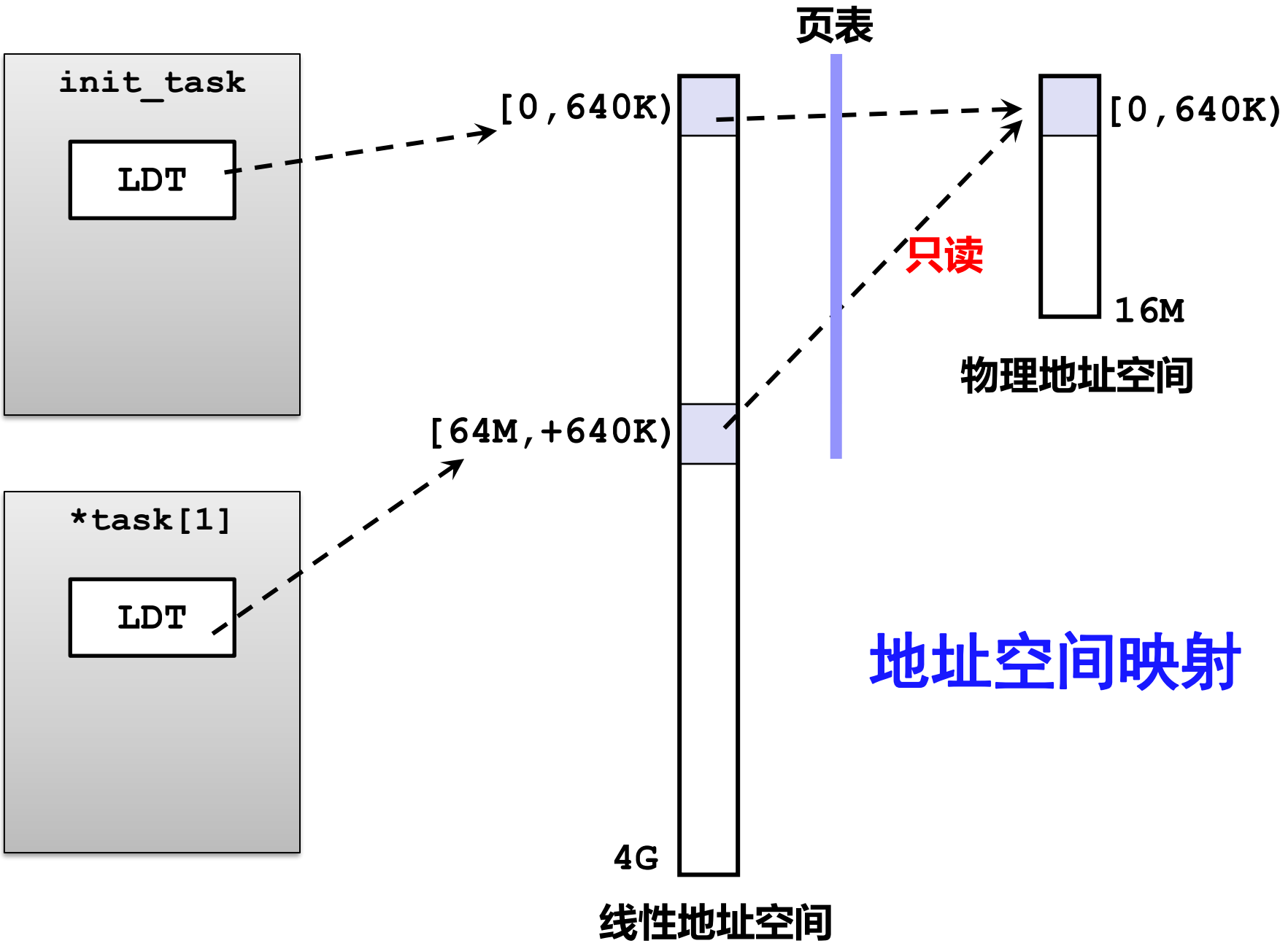

0x0017
DS
LDTR
TR
GDTR
寄存器
内存
7#
6#
1#
0#
GDT
1号进程数据段
0x04000000
*task[1]
LDT
2#
1#
0#
TSS

1号进程的变量mynext在内存的什么位置？

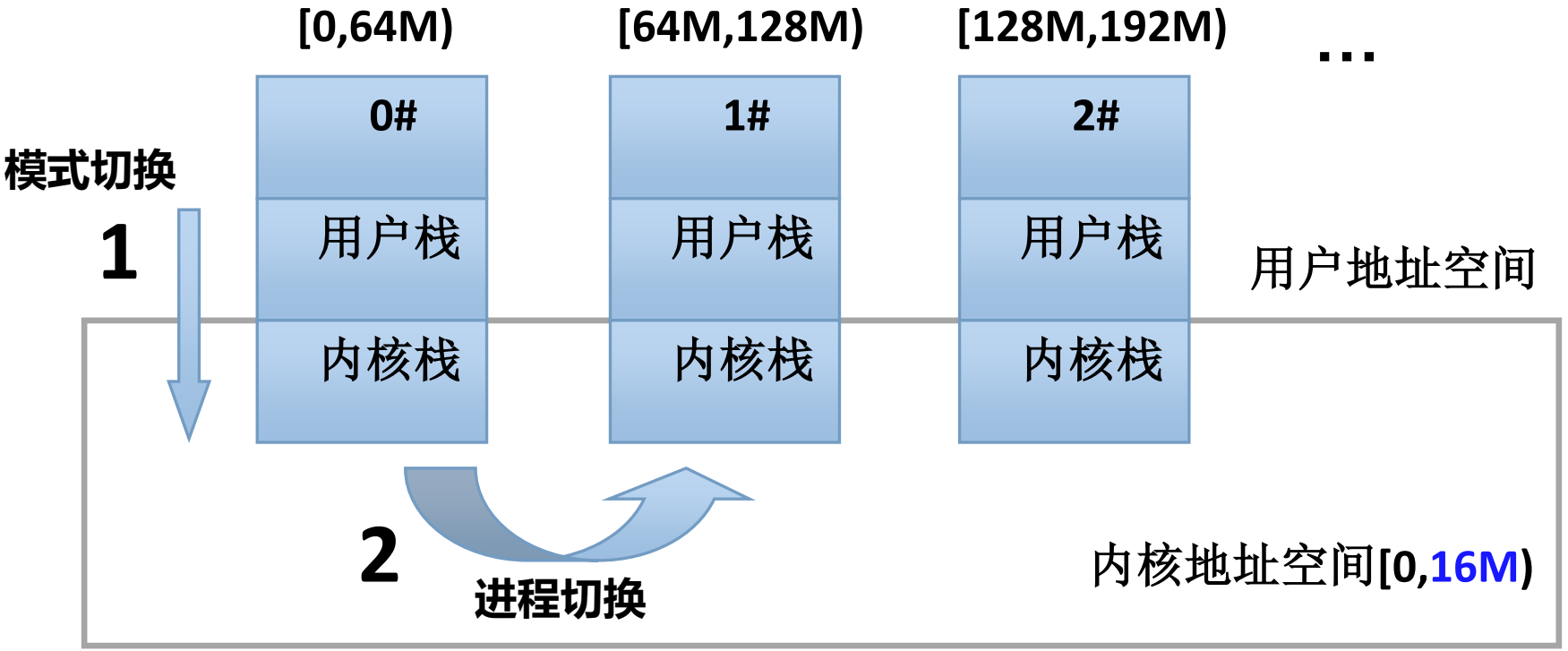


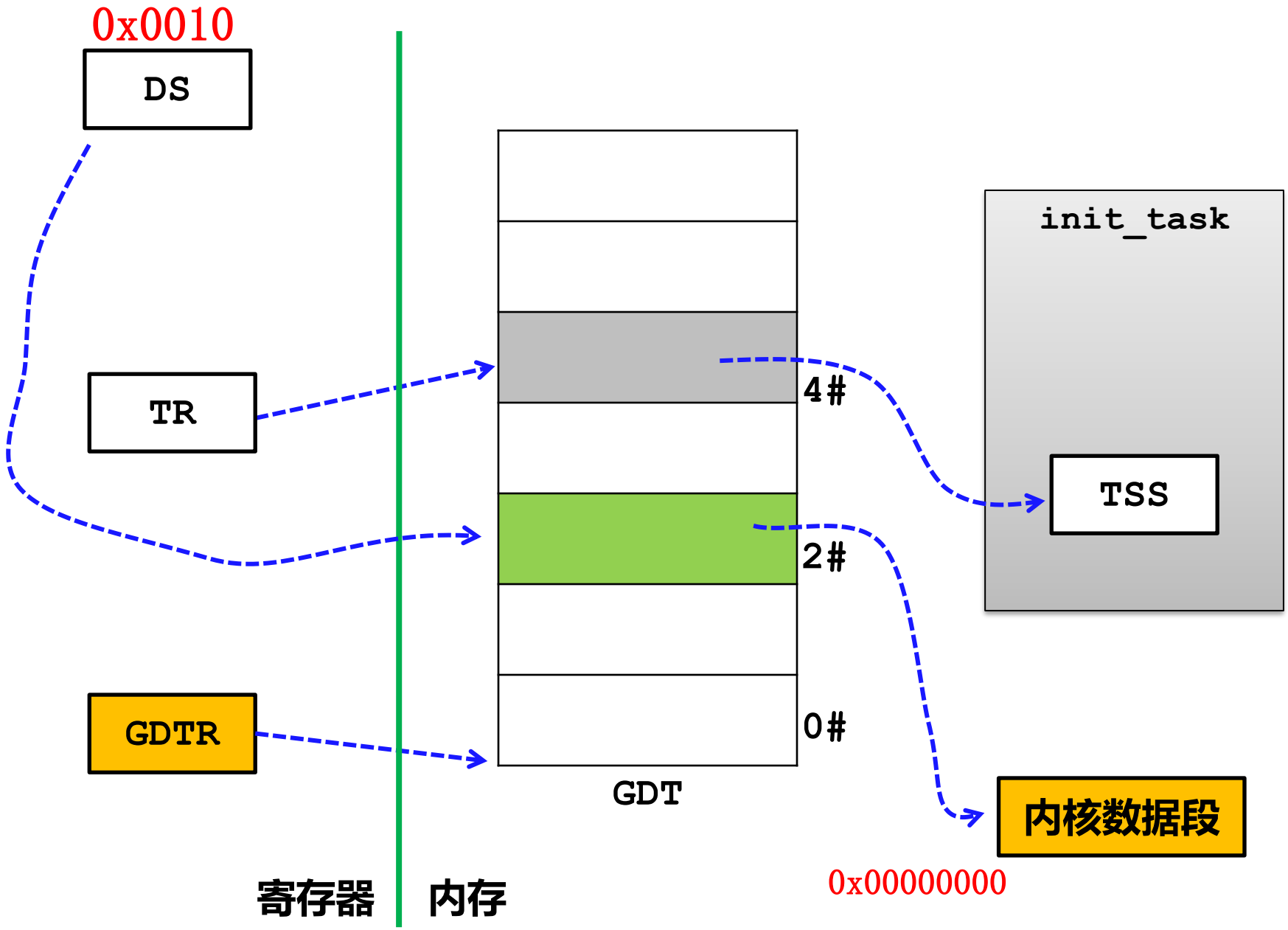

*task[1]
LDT
TSS
7#
6#
5#
4#
1#
0#
LDTR
TR
GDTR
init_task
LDT
TSS
寄存器
内存
GDT


线性地址空间[64M, +640K)
*task[1]
LDT
TSS
7#
6#
LDTR
5#
TR
4#
init_task
LDT
线性地址空间[0, +640K)
TSS
0#
GDTR
寄存器
内存
GDT


页表
init_task
LDT
[0,640K)
[0,640K)
只读
16M
物理地址空间
[64M,+640K)
*task[1]
LDT
地址空间映射
4G
线性地址空间

线性地址空间分配


[0,64M)
[64M,128M)
[128M,192M)
...
0#
1#
2#
模式切换
1
用户栈
用户栈
用户栈
用户地址空间
内核栈
内核栈
内核栈
2
进程切换
内核地址空间[0,16M)


0x0010
DS
TR
GDTR
4#
2#
0#
GDT
init_task
TSS
内核数据段
0x00000000
寄存器
内存

填空题 3分

设置

1号进程执行到sys_pause时，默认数据段（DS）的基地址是多少？ [填空1]

正常使用填空题需3.0以上版本雨课堂

作答

小结

一. 连续空间分配

多道连续可变分区法

二. 不连续空间分配

段式管理、页式管理、段页式管理、

“进程创建”中的内存拷贝

三. 页式虚存管理

小结

- 段式管理：
  - ① 1.3版内核
  - ② 段式管理基本原理
  - ③ 演示：1号进程的mynext变量地址
  - ④ 实验练习：0号进程的mynext变量
  - ⑤ 0号进程和1号进程的虚空间分配