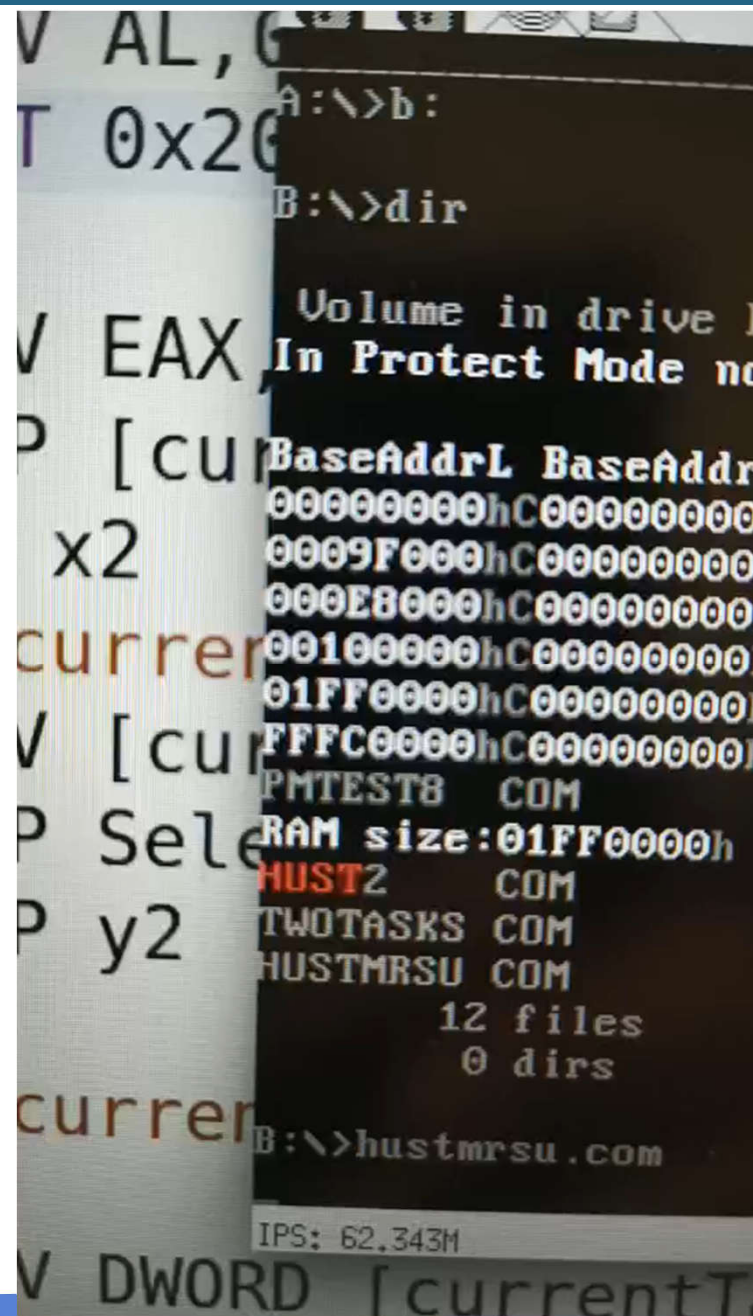


课程设计解释和提示【2022.01.27】苏曙光老师

● 任务一

- 启动保护模式，建立两个任务
（两个任务分别循环输出
“**HUST**”和“**IS19**”字符串）

- 示例：**HUST** 和 **MRSU**



```
AL, 0x20
A:\>b:
B:\>dir

Volume in drive B
In Protect Mode no

BaseAddrL BaseAddr
00000000hC00000000
0009F000hC00000000
000E8000hC00000000
00100000hC00000000
01FF0000hC00000000
FFFC0000hC00000000
PMTEST8.COM
RAM size:01FF0000h
HUST2.COM
TWOTASKS.COM
HUSTMRSU.COM
12 files
0 dirs

B:\>hustmrsu.com

IPS: 62.343M
V DWORD [currentT
```

● 任务一[书上类似的示例]

```
A:\>b:\pctest8.com
```

```
A:\>_
```

《一个操作系统的实现》
chapter3/h/pctest8.asm

```
In Protect Mode now. ^_^
```

BaseAddrL	BaseAddrH	LengthLow	LengthHigh	Type
00000000h	00000000h	0009FC00h	00000000h	00000001h
0009FC00h	00000000h	00000400h	00000000h	00000002h
000EB000h	00000000h	00018000h	00000000h	00000002h
00100000h	00000000h	01F00000h	00000000h	00000001h
FFFC0000h	00000000h	00040000h	00000000h	00000002h

```
RAM size:02000000h
```

```
Foo  
Bar
```

● 任务一[书上类似的示例]

```
409  foo:
410  OffsetFoo          equ    foo - $$
411      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
412      mov    al, 'F'
413      mov    [gs:((80 * 17 + 0) * 2)], ax ; 屏幕第 17 行, 第 0 列。
414      mov    al, 'o'
415      mov    [gs:((80 * 17 + 1) * 2)], ax ; 屏幕第 17 行, 第 1 列。
416      mov    [gs:((80 * 17 + 2) * 2)], ax ; 屏幕第 17 行, 第 2 列。
417      ret
418  LenFoo            equ    $ - foo
419
420  bar:
421  OffsetBar          equ    bar - $$
422      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
423      mov    al, 'B'
424      mov    [gs:((80 * 18 + 0) * 2)], ax ; 屏幕第 18 行, 第 0 列。
425      mov    al, 'a'
426      mov    [gs:((80 * 18 + 1) * 2)], ax ; 屏幕第 18 行, 第 1 列。
427      mov    al, 'r'
428      mov    [gs:((80 * 18 + 2) * 2)], ax ; 屏幕第 18 行, 第 2 列。
429      ret
430  LenBar            equ    $ - bar
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

● 任务一[书上类似的示例]

```
339      call    SetupPaging          ; 启动分页
340
341      call    SelectorFlatC:ProcPagingDemo
342      call    PSwitch              ; 切换页目录, 改变地址映射关系
343      call    SelectorFlatC:ProcPagingDemo
344
345      ret
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

```
329      push    LenPagingDemoAll
330      push    OffsetPagingDemoProc
331      push    ProcPagingDemo
332      call    MemCpy
333      add     esp, 12
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

```
402 PagingDemoProc:
403 OffsetPagingDemoProc equ PagingDemoProc - $$
404      mov     eax, LinearAddrDemo
405      call    eax
406      retf
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

```
13 LinearAddrDemo equ 00401000h
14 ProcFoo         equ 00401000h
15 ProcBar         equ 00501000h
16 ProcPagingDemo  equ 00301000h
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

● 任务一[书上类似的示例]

```
409  foo:
410  OffsetFoo          equ    foo - $$
411      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
412      mov    al, 'F'
413      mov    [gs:((80 * 17 + 0) * 2)], ax ; 屏幕第 17 行, 第 0 列。
414      mov    al, 'o'
415      mov    [gs:((80 * 17 + 1) * 2)], ax ; 屏幕第 17 行, 第 1 列。
416      mov    [gs:((80 * 17 + 2) * 2)], ax ; 屏幕第 17 行, 第 2 列。
417      ret
418  LenFoo            equ    $ - foo
419
420  bar:
421  OffsetBar          equ    bar - $$
422      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
423      mov    al, 'B'
424      mov    [gs:((80 * 18 + 0) * 2)], ax ; 屏幕第 18 行, 第 0 列。
425      mov    al, 'a'
426      mov    [gs:((80 * 18 + 1) * 2)], ax ; 屏幕第 18 行, 第 1 列。
427      mov    al, 'r'
428      mov    [gs:((80 * 18 + 2) * 2)], ax ; 屏幕第 18 行, 第 2 列。
429      ret
430  LenBar            equ    $ - bar
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

● 任务一[书上类似的示例]

```
8  PageDirBase      equ      200000h ; 页目录开始地址: 2M
9  PageTblBase      equ      201000h ; 页表开始地址: 2M+4K
   ...
19  LABEL_DESC_PAGE_DIR: Descriptor PageDirBase, 4095, DA_DRW; Page Directory
20  LABEL_DESC_PAGE_TBL: Descriptor PageTblBase, 1023, DA_DRW|DA_LIMIT_4K; Page Tables
   ...
34  SelectorPageDir  equ      LABEL_DESC_PAGE_DIR      - LABEL_GDT
35  SelectorPageTbl  equ      LABEL_DESC_PAGE_TBL      - LABEL_GDT
   ...
202 ; 启动分页机制-----
203 SetupPaging:
204     ; 为简化处理, 所有线性地址对应相等的物理地址.
205
206     ; 首先初始化页目录
207     mov     ax, SelectorPageDir      ; 此段首地址为PageDirBase
208     mov     es, ax
209     mov     ecx, 1024                ; 共1K 个表项
210     xor     edi, edi
211     xor     eax, eax
212     mov     eax, PageTblBase | PG_P | PG_USU | PG_RWW
213 .1:
214     stosd
215     add     eax, 4096                ; 为了简化, 所有页表在内存中是连续的.
216     loop    .1
```

《一个操作系统的实现》
chapter3/f/pmtest6.asm

● 任务一[书上类似的示例]

```
230      mov     eax, PageDirBase
231      mov     cr3,  eax
232      mov     eax,  cr0
233      or      eax, 80000000h
234      mov     cr0,  eax
```

《一个操作系统的实现》
chapter3/f/pmtest6.asm



图3.30 cr3

《一个操作系统的实现》
chapter3/f/pmtest6.asm

cr3又叫做PDBR（Page-Directory Base Register）。它的高20位将是页目录表首地址的高20位，页目录表首地址的低12位会是零，也就是说，页目录表会是4KB对齐的。类似地，PDE中的页表基址（Page-Table Base Address）以及PTE中的页基址（Page Base Address）也是用高20位来表示4KB对齐的页表和页。

● 任务一[书上类似的示例]

```
230      mov     eax, PageDirBase
231      mov     cr3, eax
232      mov     eax, cr0
233      or      eax, 80000000h
234      mov     cr0, eax
```

《一个操作系统的实现》
chapter3/f/pmtest6.asm

```
339      call    SetupPaging           ; 启动分页
340
341      call    SelectorFlatC:ProcPagingDemo
342      call    PSwitch                ; 切换页目录, 改变地址映射关系
343      call    SelectorFlatC:ProcPagingDemo
344
345      ret
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm



● 任务一[书上类似的示例]

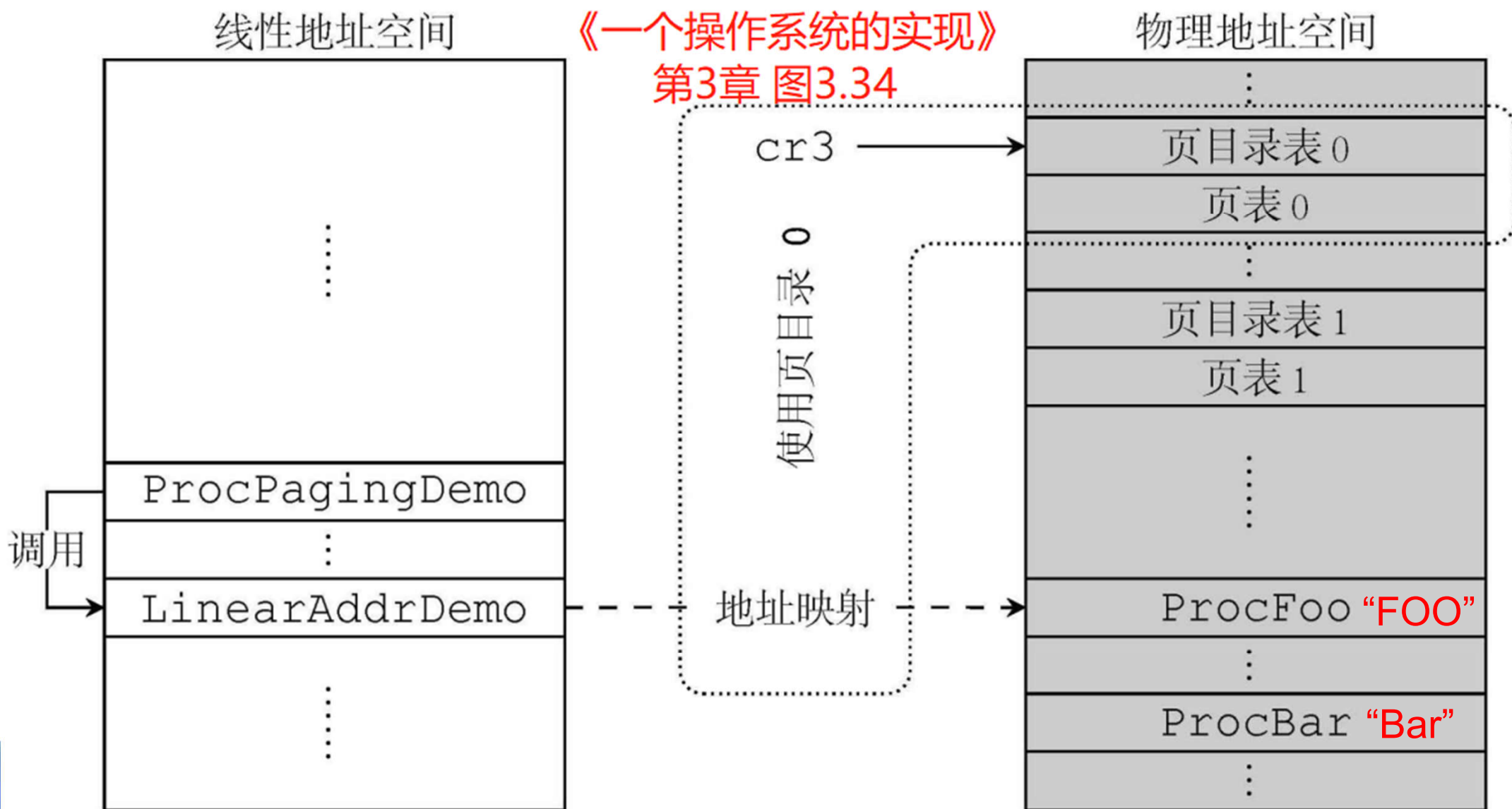


图3.34 开始时的内存映射关系

● 任务一[书上类似的示例]

```
A:\>b:\pctest8.com
```

```
A:\>_
```

《一个操作系统的实现》
chapter3/h/pctest8.asm

```
In Protect Mode now. ^_^
```

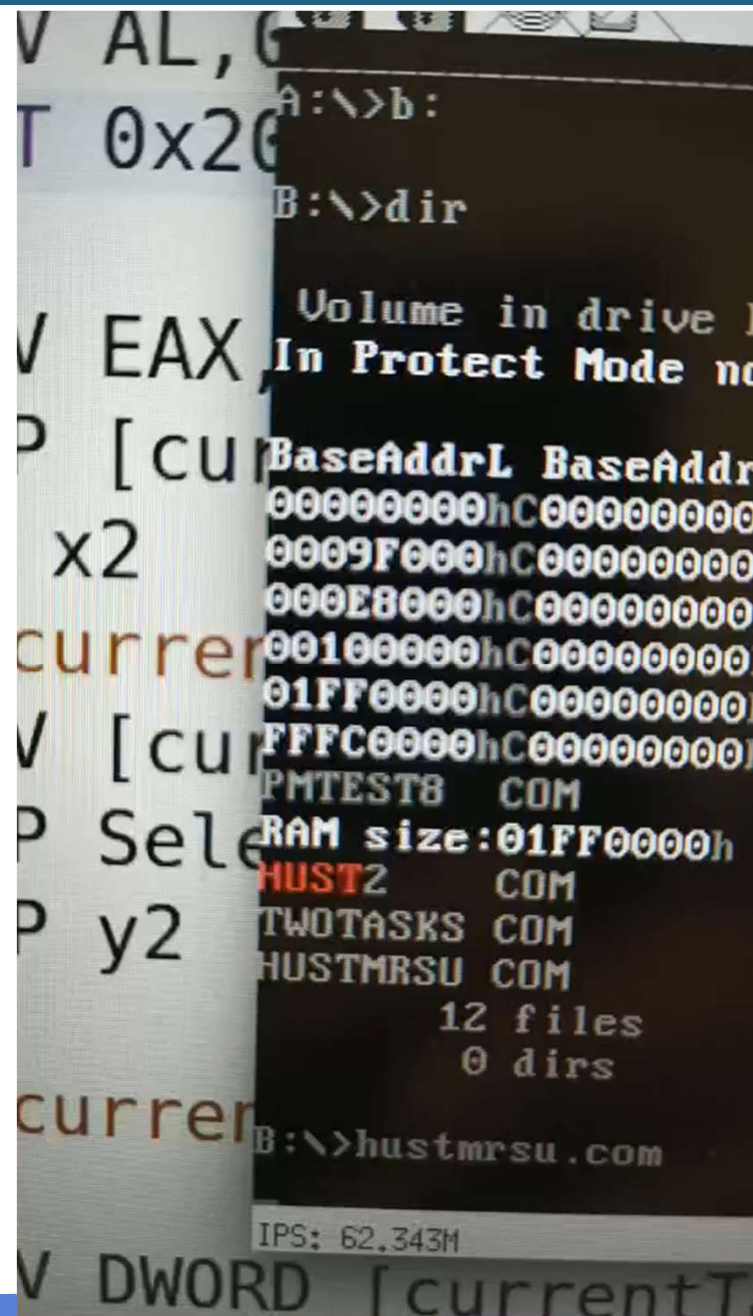
BaseAddrL	BaseAddrH	LengthLow	LengthHigh	Type
00000000h	00000000h	0009FC00h	00000000h	00000001h
0009FC00h	00000000h	00000400h	00000000h	00000002h
000EB000h	00000000h	00018000h	00000000h	00000002h
00100000h	00000000h	01F00000h	00000000h	00000001h
FFFC0000h	00000000h	00040000h	00000000h	00000002h

```
RAM size:02000000h
```

```
Foo  
Bar
```

● 任务一

- 启动保护模式，建立两个任务
（两个任务分别循环输出
“**HUST**”和“**IS19**”字符串）
- 示例：**HUST** 和 **MRSU**



```
AL, 0x20
T 0x20
A:\>b:
B:\>dir

Volume in drive B
In Protect Mode no

BaseAddrL BaseAddr
00000000hC00000000
0009F000hC00000000
000E8000hC00000000
00100000hC00000000
01FF0000hC00000000
FFFC0000hC00000000
PMTEST8.COM
RAM size:01FF0000h
HUST2.COM
TWOTASKS.COM
HUSTMRSU.COM
12 files
0 dirs

B:\>hustmrsu.com

IPS: 62.343M
V DWORD [currentT
```


● 任务一[书上类似的示例]

```
409  foo:
410  OffsetFoo          equ    foo - $$
411      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
412      mov    al, 'F'
413      mov    [gs:((80 * 17 + 0) * 2)], ax ; 屏幕第 17 行, 第 0 列。
414      mov    al, 'o'
415      mov    [gs:((80 * 17 + 1) * 2)], ax ; 屏幕第 17 行, 第 1 列。
416      mov    [gs:((80 * 17 + 2) * 2)], ax ; 屏幕第 17 行, 第 2 列。
417      ret
418  LenFoo             equ    $ - foo
419
420  bar:
421  OffsetBar          equ    bar - $$
422      mov    ah, 0Ch                ; 0000: 黑底      1100: 红字
423      mov    al, 'B'
424      mov    [gs:((80 * 18 + 0) * 2)], ax ; 屏幕第 18 行, 第 0 列。
425      mov    al, 'a'
426      mov    [gs:((80 * 18 + 1) * 2)], ax ; 屏幕第 18 行, 第 1 列。
427      mov    al, 'r'
428      mov    [gs:((80 * 18 + 2) * 2)], ax ; 屏幕第 18 行, 第 2 列。
429      ret
430  LenBar             equ    $ - bar
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

● 任务一[书上类似的示例]

```
409  foo:
410  OffsetFoo          equ    foo - $$
411      mov    ah, 0Ch  "HUST"          ; 0000: 黑底      1100: 红字
412      mov    al, 'F'
413      mov    [gs:((80 * 17 + 0) * 2)], ax ; 屏幕第 17 行, 第 0 列。
414      mov    al, 'o'
415      mov    [gs:((80 * 17 + 1) * 2)], ax ; 屏幕第 17 行, 第 1 列。
416      mov    [gs:((80 * 17 + 2) * 2)], ax ; 屏幕第 17 行, 第 2 列。
417      ret  JMP
418  LenFoo             equ    $ - foo
419
420  bar:
421  OffsetBar          equ    bar - $$
422      mov    ah, 0Ch  "IS19"          ; 0000: 黑底      1100: 红字
423      mov    al, 'B'
424      mov    [gs:((80 * 18 + 0) * 2)], ax ; 屏幕第 18 行, 第 0 列。
425      mov    al, 'a'
426      mov    [gs:((80 * 18 + 1) * 2)], ax ; 屏幕第 18 行, 第 1 列。
427      mov    al, 'r'
428      mov    [gs:((80 * 18 + 2) * 2)], ax ; 屏幕第 18 行, 第 2 列。
429      ret  JMP
430  LenBar             equ    $ - bar
```

《一个操作系统的实现》
chapter3/h/pmtest8.asm

● 任务一

```
409  foo:
410  OffsetFoo          equ    foo - $$
411      mov    ah, 0Ch  "HUST"          ; 0000: 黑底      1100: 红字
412      mov    al, 'F'
413      mov    [gs:((80 * 17 + 0) * 2)], ax ; 屏幕第 17 行, 第 0 列。
414      mov    al, 'o'
415      mov    [gs:((80 * 17 + 1) * 2)], ax ; 屏幕第 17 行, 第 1 列。
416      mov    [gs:((80 * 17 + 2) * 2)], ax ; 屏幕第 17 行, 第 2 列。
417      ret  JMP
418  LenFoo             equ    $ - foo
419
420  bar:
421  OffsetBar          equ    bar - $$
422      mov    ah, 0Ch  "IS19"          ; 0000: 黑底      1100: 红字
423      mov    al, 'B'
424      mov    [gs:((80 * 18 + 0) * 2)], ax ; 屏幕第 18 行, 第 0 列。
425      mov    al, 'a'
426      mov    [gs:((80 * 18 + 1) * 2)], ax ; 屏幕第 18 行, 第 1 列。
427      mov    al, 'r'
428      mov    [gs:((80 * 18 + 2) * 2)], ax ; 屏幕第 18 行, 第 2 列。
429      ret  JMP
430  LenBar             equ    $ - bar
```

《一个操作系统的实现》
chatper3/h/pmtest8.asm

● 任务一

■ 启动保护模式，建立两个任务（两个任务分别循环输出“**HUST**”和“**IS19**”字符串）

■ 示例：**HUST** 和 **MRSU**

■ 思路

◆ 两个过程，两套页目录/页表

◆ 两个任务

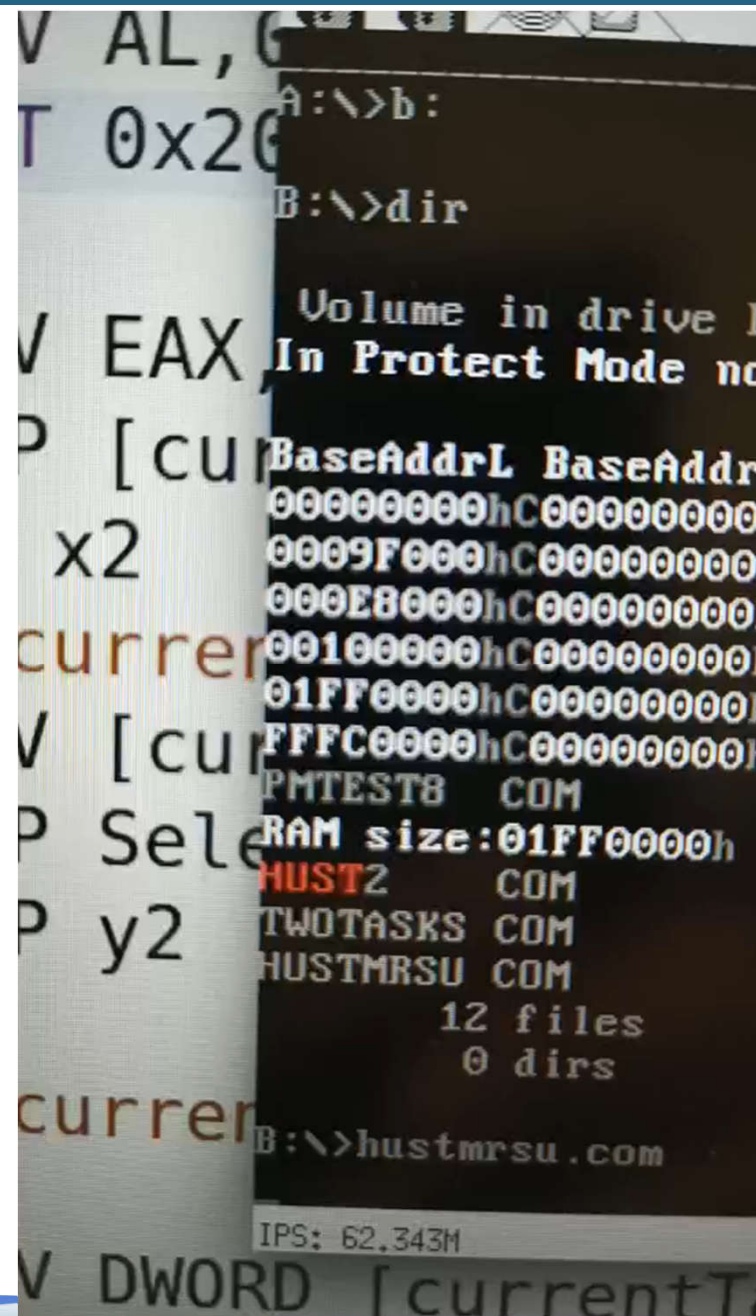
◆ 任务LDT + TSS

◆ 按时间片切换两个任务

□ 初始化8253时钟

□ 初始化8259中断

□ 时钟中断服务程序切换任务



```
AL, 0x20
A:\>b:
B:\>dir

Volume in drive B
In Protect Mode no

BaseAddrL BaseAddr
00000000hC00000000
0009F000hC00000000
000E8000hC00000000
00100000hC00000000
01FF0000hC00000000
FFFC0000hC00000000
PMTEST8.COM
RAM size:01FF0000h
HUST2.COM
TWO TASKS.COM
HUSTMRSU.COM
12 files
0 dirs
B:\>hustmrsu.com

IPS: 62.343M
V DWORD [currentT
```

课程设计解释和提示【2022.01.27】苏曙光老师

● 任务一

定义GDT和段描述符;
定义选择子;
获取内存信息;
完善段描述符;
修改CR0进入保护模式;

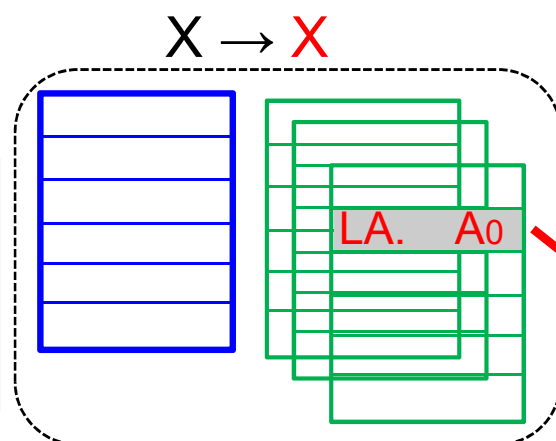
准备Task0和Task1代码;
准备页目录0和页表;
准备页目录1和页表;

```
MOV CR3, PT0_base  
CALL LinearAddrDemo
```

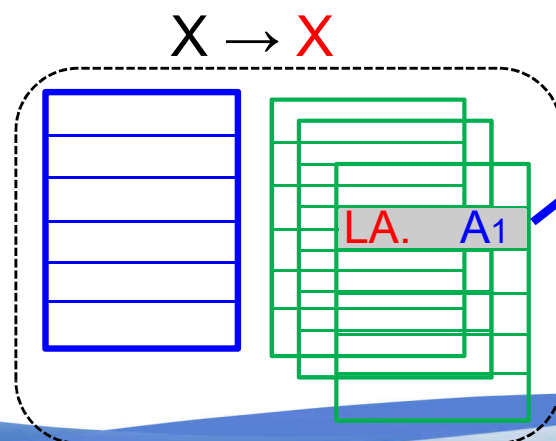
```
MOV CR3, PT1_base  
CALL LinearAddrDemo
```

时钟中断服务程序（50ms间隔）：
{ □ | □ }

定义IDT与中断门描述符;
初始化8253芯片;
初始化8259芯片



A0:
Task0: While(TRUE) { 输出HUST }



A1:
Task1: While(TRUE) { 输出IS19 }

课程设计解释和提示【2022.01.27】苏曙光老师

● 任务一[预备实验：两个任务，仅输出1次]

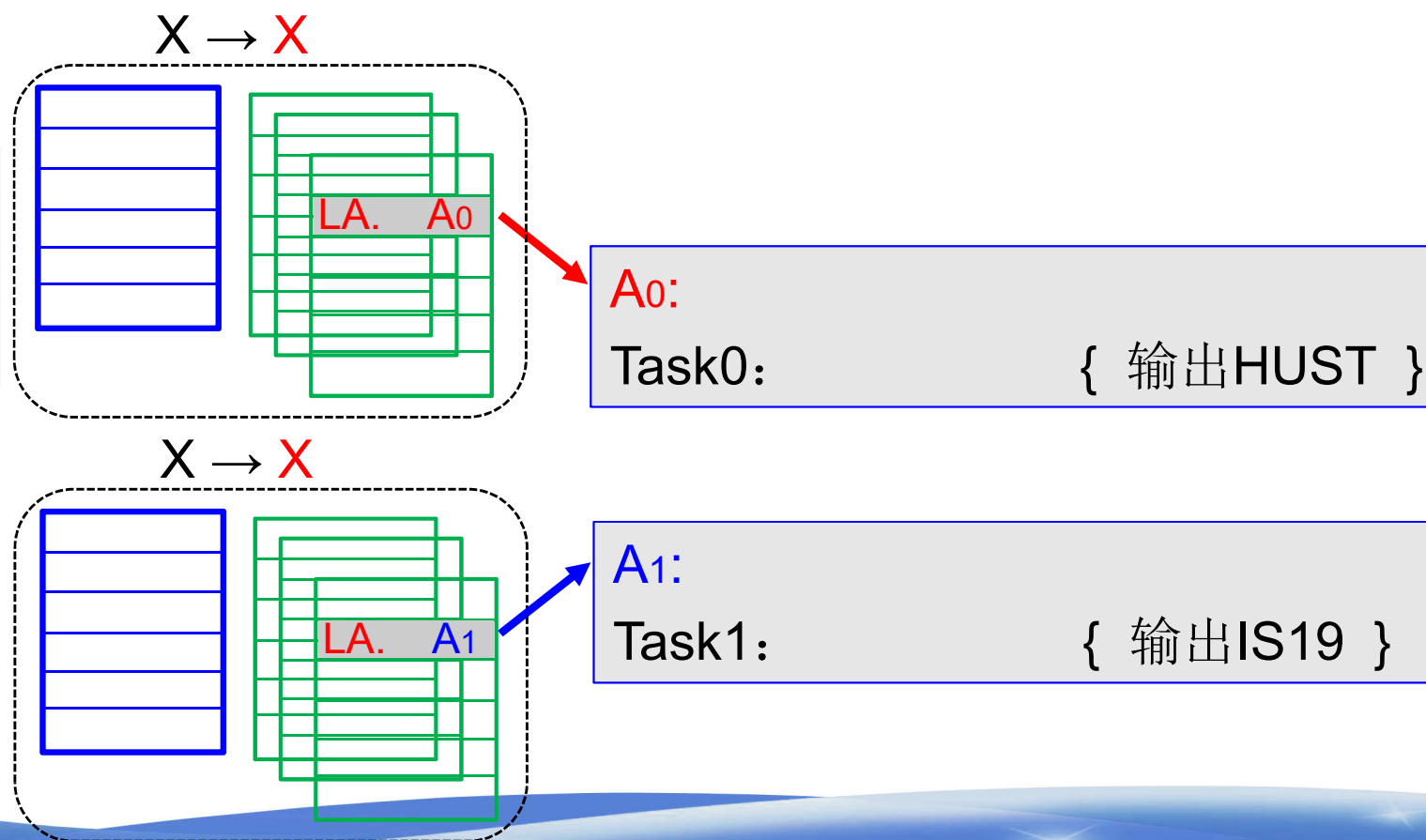
定义GDT和段描述符;
定义选择子;
获取内存信息;
完善段描述符;
修改CR0进入保护模式;

准备Task0和Task1代码;
准备页目录0和页表;
准备页目录1和页表;

```
MOV CR3, PT0_base  
CALL LinearAddrDemo
```

```
MOV CR3, PT1_base  
CALL LinearAddrDemo
```

ch3/h/pmtest8.asm



- 任务一（参考《于渊的书前三章》和苏老师课件）
 - (0) 环境：Ubuntu + Bochs + freedos + NASM（参考书）
 - (1) 定义段和描述符表
 - (2) 初始化描述符表
 - (3) 初始化选择子
 - (4) 定义16位的各功能段
 - (5) 定义32位的各功能段，包括LDT段
 - (6) 进入保护模式
 - (7) 初始化页表：可以简单地做线性映射
 - (8) 定义两个任务：两个TSS，两个LDT，两套页表
 - (9) 利用时钟中断切换两个任务
 - (10) 任务：简单输出“HUST”或“IS19”

- 任务一（参考《于渊的书前三章》和苏老师课件）

保护模式下的任务/进程

- 任务由两个部分组成

苏老师课件第7章7.5节，群里面有

- 任务执行环境

- ◆ 代码段

- ◆ 数据段

- ◆ 堆栈段

- 每一个特权级都有一个堆栈段

- ◆ LDT/LDTR/LLDT

- 构成一个局部地址空间，不能被其他任务访问。

- 任务状态段TSS（Task State Segment）

- ◆ 保存任务状态信息

- ◆ TSS描述符

- ◆ TR/LTR

课程设计解释和提示【2022.01.27】苏曙光老师

- 任务一（参考《于渊的书前三章》和苏老师课件）

网安19级操作系统课程群

聊天 公告 相册 文件 应用 设置

共33个文件 (已使用103MB/10GB)

课设任务一的基础知识请参考
苏老师版本课件的7.5节约50页

文件	上传时间	过期时间	大小	上传者	下载次数
复习版本-第3章.ppt					
复习版本-第2章.pdf	2021-12-09	永久	1.19MB	苏曙光...	405次
复习版本-第1章.pdf	2021-12-09	永久	3.88MB	苏曙光...	418次
第4-6章 课堂习题和解答.pdf	2021-12-09	永久	456KB	苏曙光...	215次
第7章 课堂习题和解答.pdf	2021-12-09	永久	429KB	苏曙光...	212次
第07章 主存管理（信安1-4班版本）.pdf	2021-12-06	永久	4.72MB	苏曙光...	99次
第7章 主存管理.ppt	2021-12-04	永久	2.69MB	真是废...	73次
操作系统原理实验-LinuxWindows综合版本 - 实验三....	2021-12-01	永久	255KB	苏曙光...	524次
操作系统原理实验-LinuxWindows综合版本 - 实验二....	2021-11-25	永久	265KB	苏曙光...	326次

- 任务一（参考《于渊的书前三章》和苏老师课件）

实模式与保护模式的内存寻址对比

苏老师课件第7章7.5节，群里面有

MOV BX, 17H

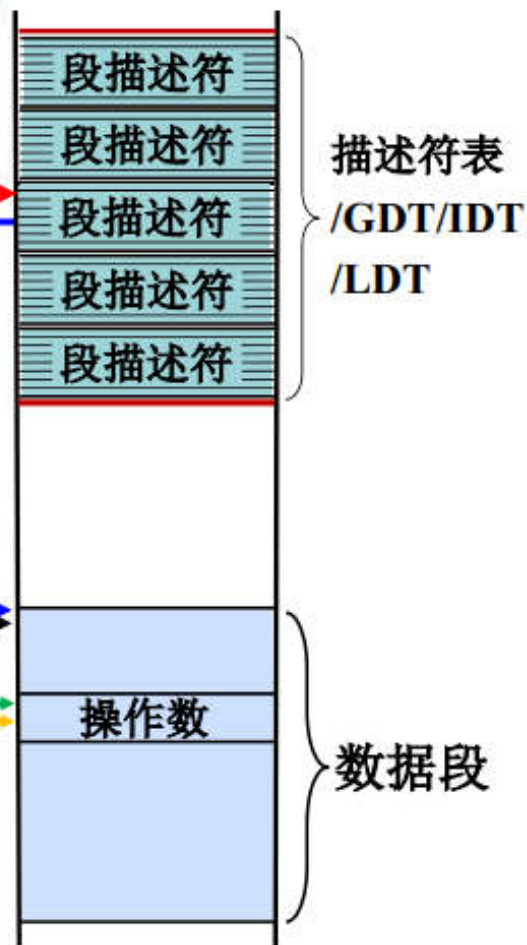
MOV DS, BX ;DS = 0001 0111

MOV AX, [100]

段寄存器：偏移量
逻辑地址 DS AX

段基址	限长
存取属性	是否在内存

描述符表寄存器/GDTR/LDTR



课程设计解释和提示【2022.01.27】苏曙光老师

● 课设任务二

- 编写设备驱动程序，对内存缓冲区进行读写
- 熟悉Linux设备驱动程序开发过程
- 实现设备的阻塞和非阻塞两种工作方式
- 理解和应用内核等待队列同步机制

● 课设内容

- 1.编写驱动程序，支持应用程序对内核缓冲区的读写
 - ◆ 设定内核缓冲区大小（例如32字节）
 - ◆ 缓冲区是环形缓冲区，驱动程序维护两个读写指针
 - ◆ 缓冲区按序读写，每个数据的读写不重复，不遗漏，
 - ◆ 编写若干个应用程序，循环读或写缓冲区的若干字节
 - 当缓冲区有足够的数据读就读，否则就阻塞进程，直到有足够数据可供读时才被唤醒；
 - 当缓冲区有足够的空位写就写，否者就阻塞进程，直到有足够空位可供写时才被唤醒；
 - ◆ 驱动程序内部维护缓冲区的读写，并适时阻塞或唤醒相应进程
 - ◆ 观察缓冲区变化与读/写进程的阻塞/被唤醒的同步情况。

● 任务二

■ 1.Linux下编写设备驱动程序

◆开辟缓冲区BUFFER供应用读写：**读/写不遗漏不重复**

◆实现设备的阻塞和非阻塞两种工作方式

■ 2.编写不少于2个应用尝试读/写前述设备（缓冲区）

◆Read(fd , nBytesCount)

◆Write(fd , nBytesCount)

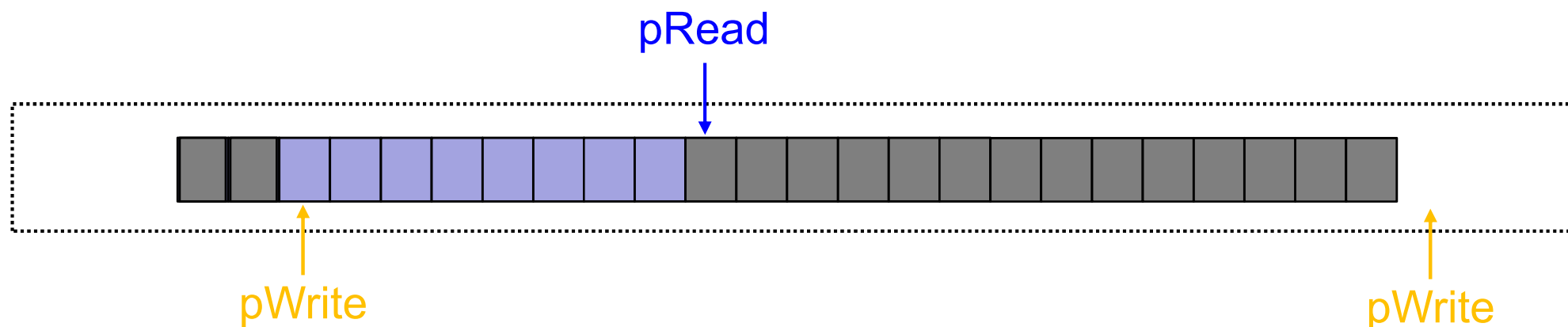
◆观察缓冲区变化与读/写进程阻塞/被唤醒的同步情况。



- 1.Linux下编写设备驱动程序

- 内设固定大小缓冲区BUFFER，**读/写不遗漏不重复**

- **读指针pRead**，**写指针pWrite**



- 2.编写不少于2个应用尝试读/写前述设备（缓冲区）

- Read(fd , nBytesCount)

- Write(fd , nBytesCount)

- 观察缓冲区变化与读/写进程阻塞/被唤醒的同步情况。

- **ps命令** 查看进程状态

- 任务二的预备知识
 - Linux驱动程序开发
 - Linux内核同步机制：等待队列，互斥锁，异步事件
 - 设备的阻塞/非阻塞工作方式
 - 进程以阻塞/非阻塞方式打开设备



寒假快乐!

