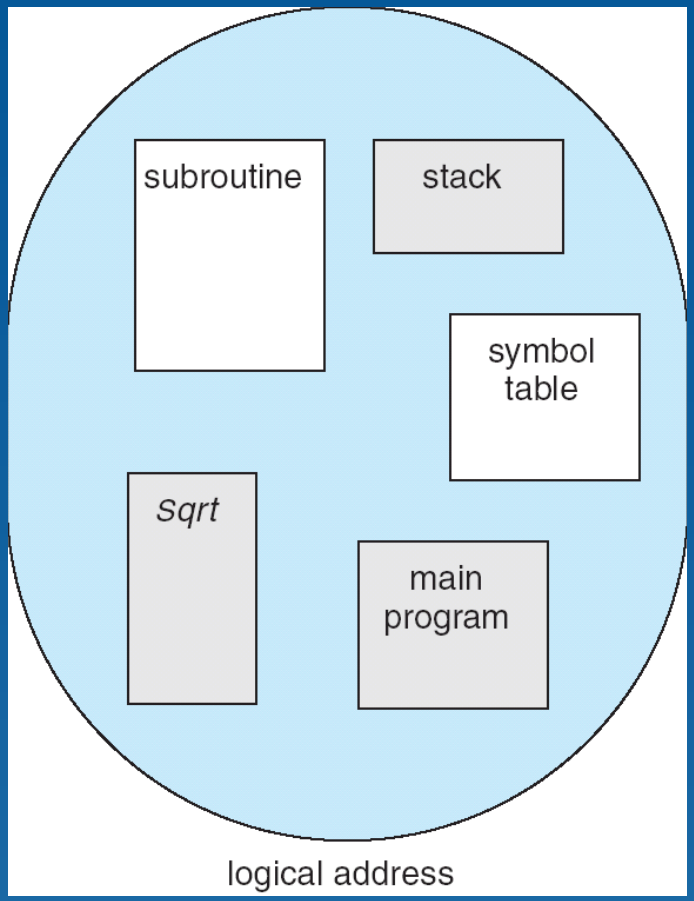
段式存储管理和示例

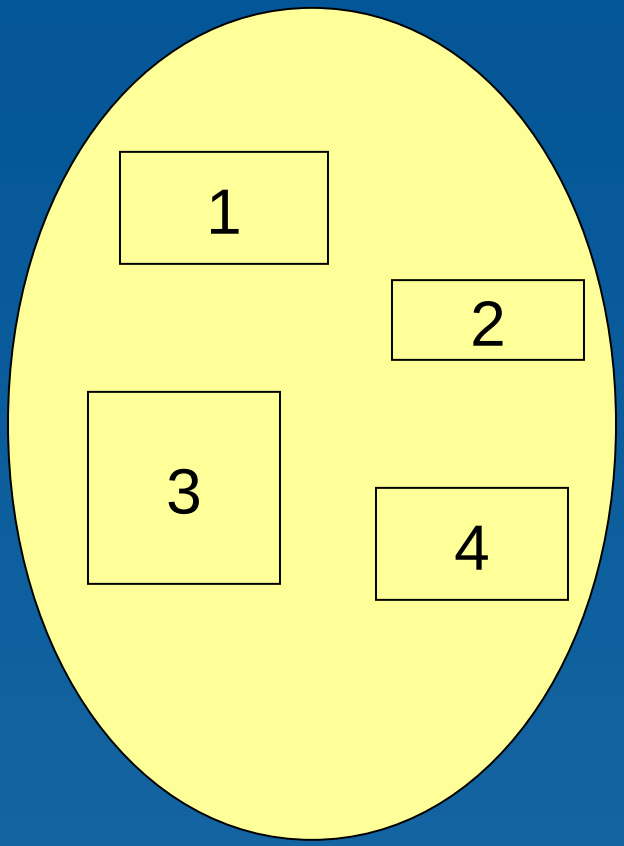
段式存储管理

- 这是一种顺应用户视角的内存管理机制
- 程序一定是由许多段代码、数据组成。“段”是自然的逻辑单元，例如:
  - main program,
  - procedure,
  - function,
  - method,
  - object,
  - local variables, global variables,
  - common block,
  - stack,
  - symbol table, arrays

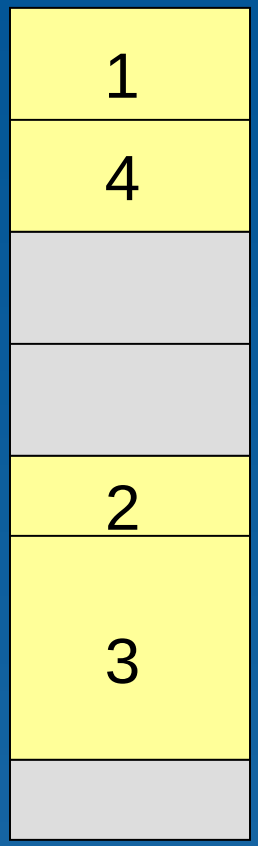
用户视角看程序


subroutine
stack
symbol table
Sqrt
main program
logical address

从逻辑地址观察“段”



用户空间



物理内存空间

段式管理的机制

- 一个逻辑地址划分成两部分:

  < 段号，段内偏移量 >

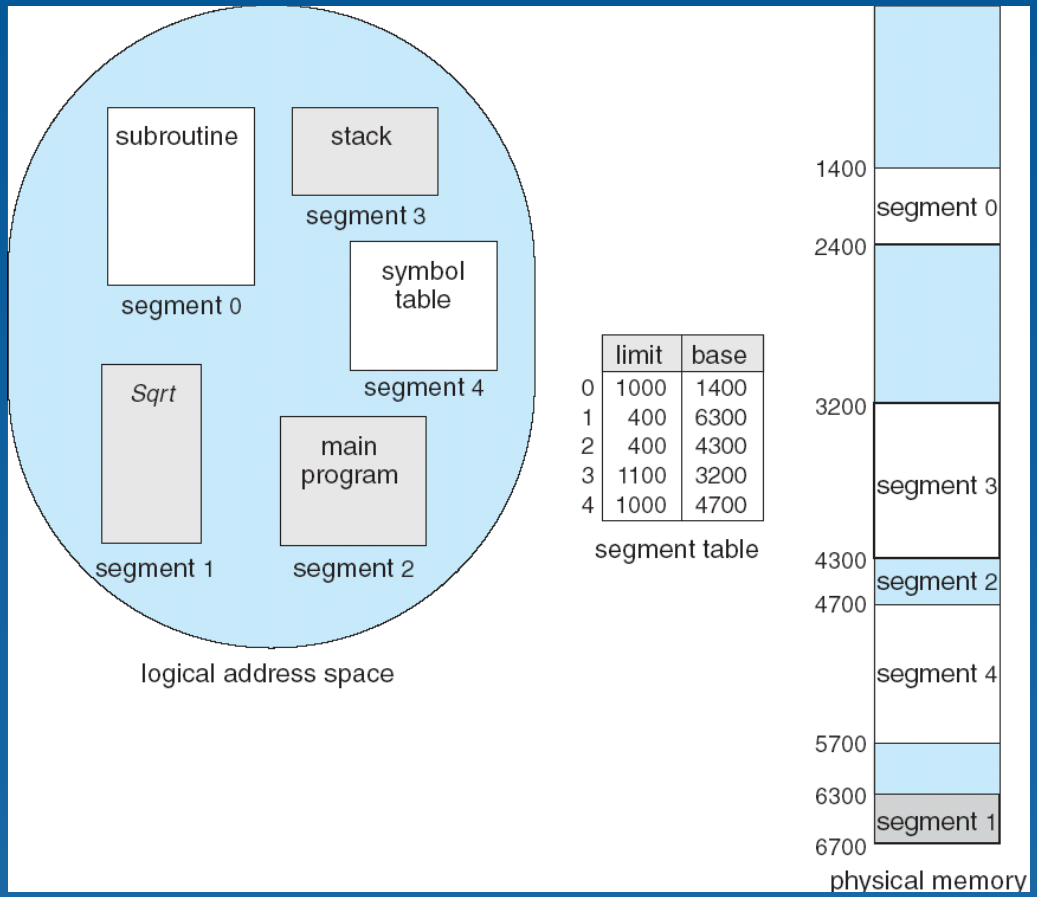
- **段表** (segment table) – 以段号为索引下标，将其映射至二维的物理地址
- 段表项内容包括
  - **基地址** (base)– 记录该“段”在物理内存的首地址
  - **界限** (limit) – 记录该“段”的长度

段式管理的机制（续）

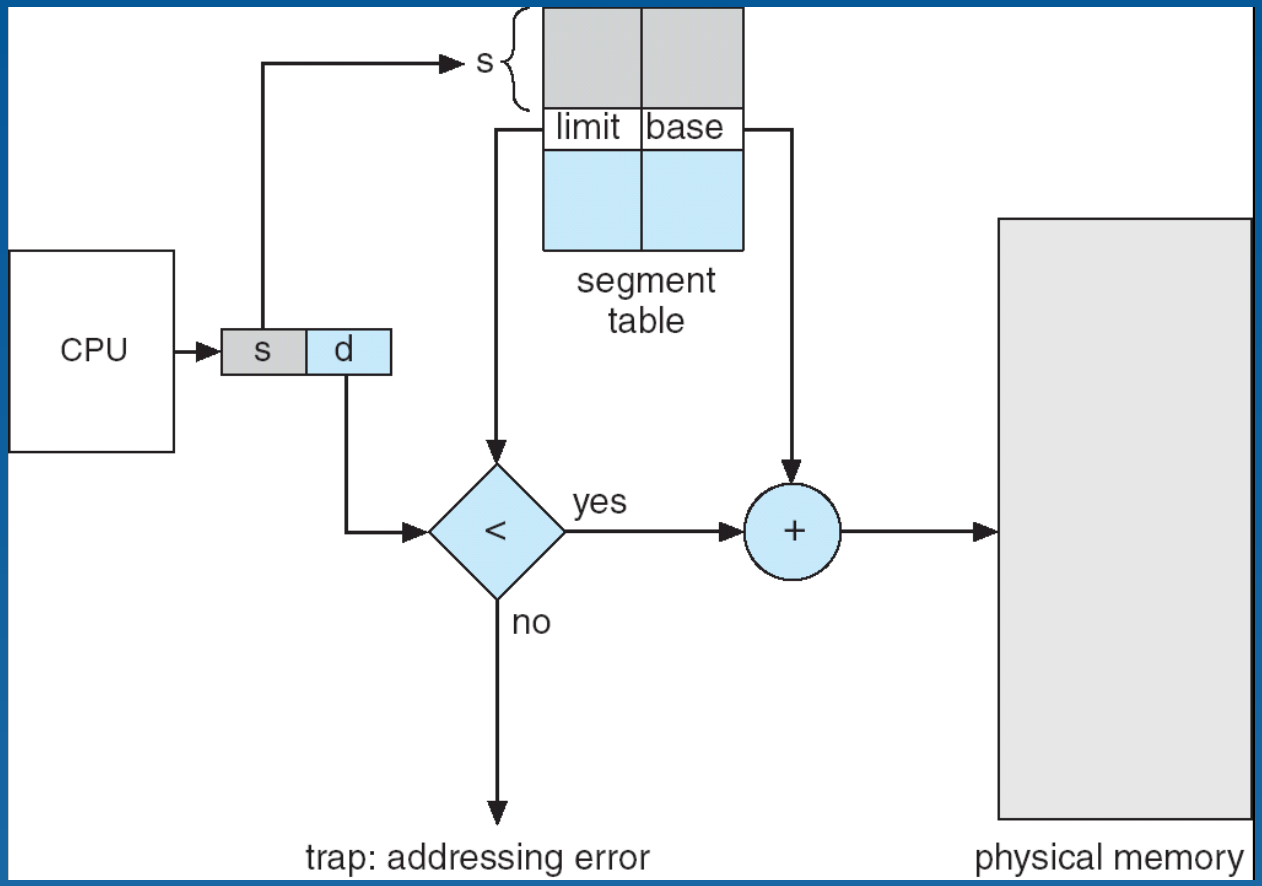
- 段表基地址寄存器 Segment-table base register (**STBR**) 指向内存中段表的首地址
- 段表长度寄存器 Segment-table length register (**STLR**) 记录程序总段数，也表示段表项的总数
- 合法的段号 ***s*** 必须满足

$$s < STLR$$

示例


subroutine
stack
segment 3
symbol table
segment 0
segment 4
Sqrt
main program
segment 1
segment 2
logical address space
limit base
0 1000 1400
1 400 6300
2 400 4300
3 1100 3200
4 1000 4700
segment table
1400
segment 0
2400
3200
segment 3
4300
segment 2
4700
segment 4
5700
6300
segment 1
6700
physical memory

地址翻译


s
limit
base
segment
table
CPU
s
d
<
yes
+
no
trap: addressing error
physical memory

段式管理机制的分析

- 内存保护
  - 每个段表项都有保护位:
    - 有效位 (1 位)，有效位 = 0 ⇒ 无效段
    - 特权位 (2 位)， read/write/execute

段式管理机制的分析（续）

- 内存分配
  - 段的长度可变（区别于页），内存分配面临 dynamic storage-allocation 问题
  - first fit/best fit/worst fit/etc.
  - 碎片， external fragmentation

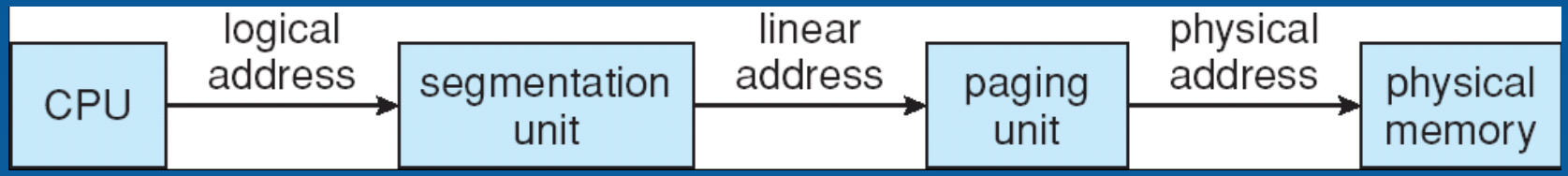
段式管理机制的分析（续）

- 重定位
  - 可以动态重定位
  - 借助段表实现
- 内存共享
  - 以“段”为最小单位，支持进程间代码共享
  - 进程必须给予共享段以相同的段号

Intel i386 的寻址特征

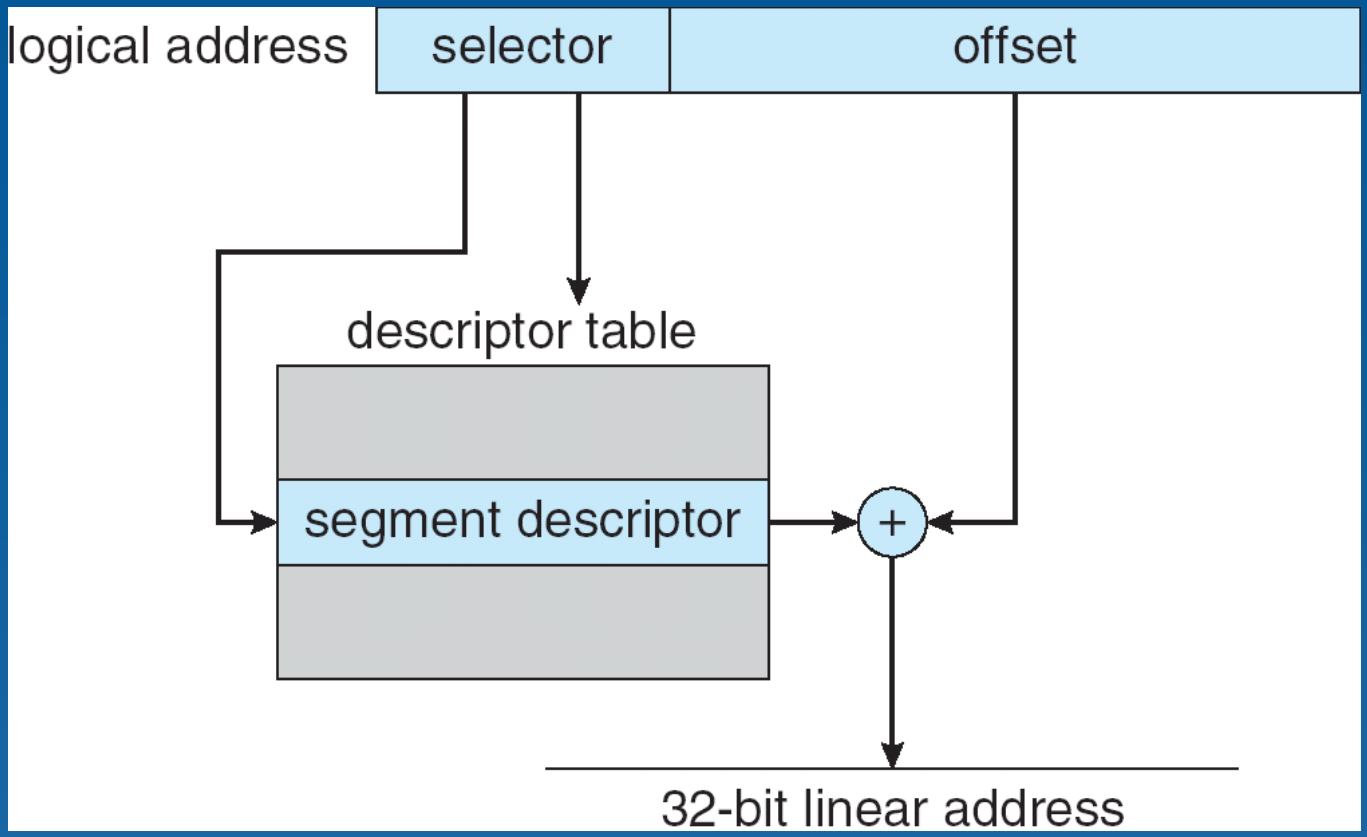
- 可以是段式存储管理；也可以是段页式存储管理（段内分页）
- CPU 产生逻辑地址
  - 逻辑地址传递给 MMU 的段式机制加以翻译
    - 期间产生线性地址
    - 实模式下，直接生成物理地址
  - 线性地址传递给 MMU 的页式机制继续翻译
    - 结果是生成了物理地址
    - 虚拟模式下，这部分才工作
    - 实模式下，这部分不工作

将逻辑地址翻译成物理地址

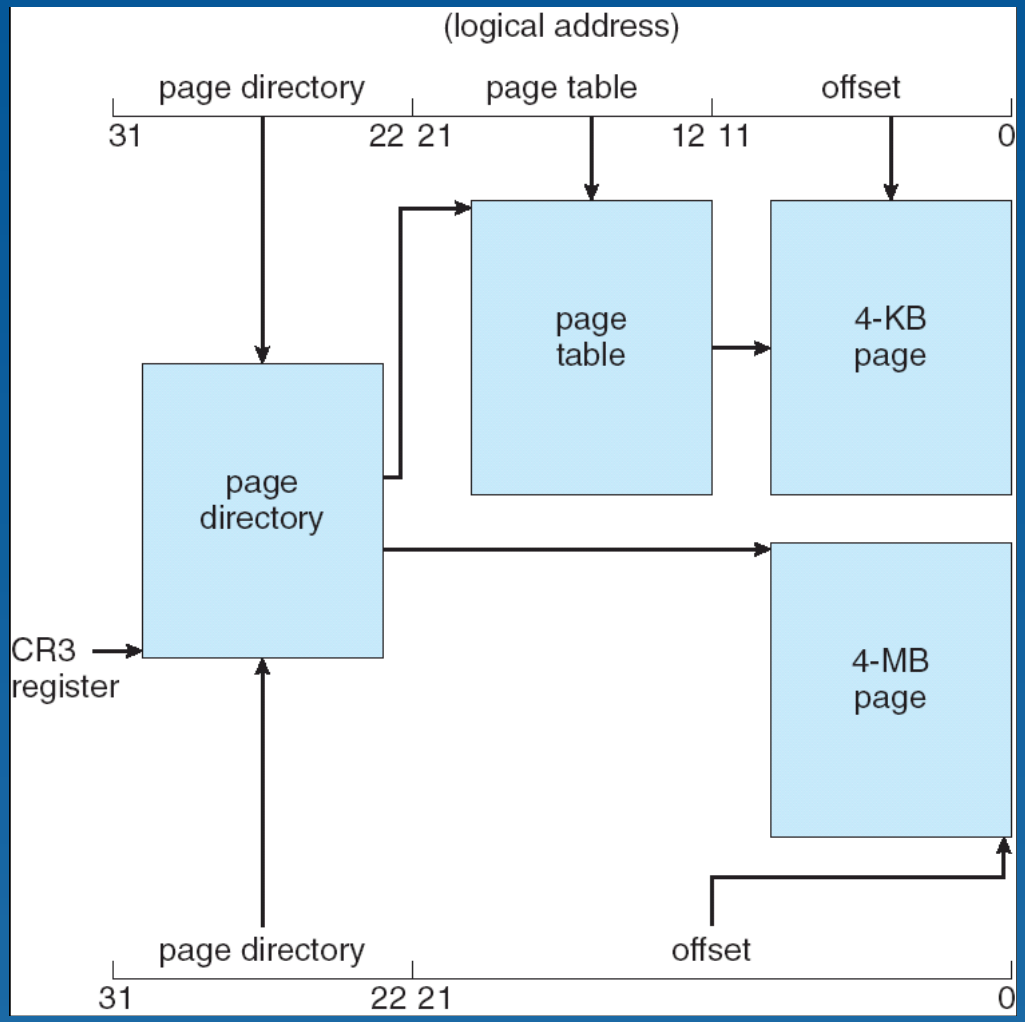


page number		page offset
p_1	p_2	d
10	10	12

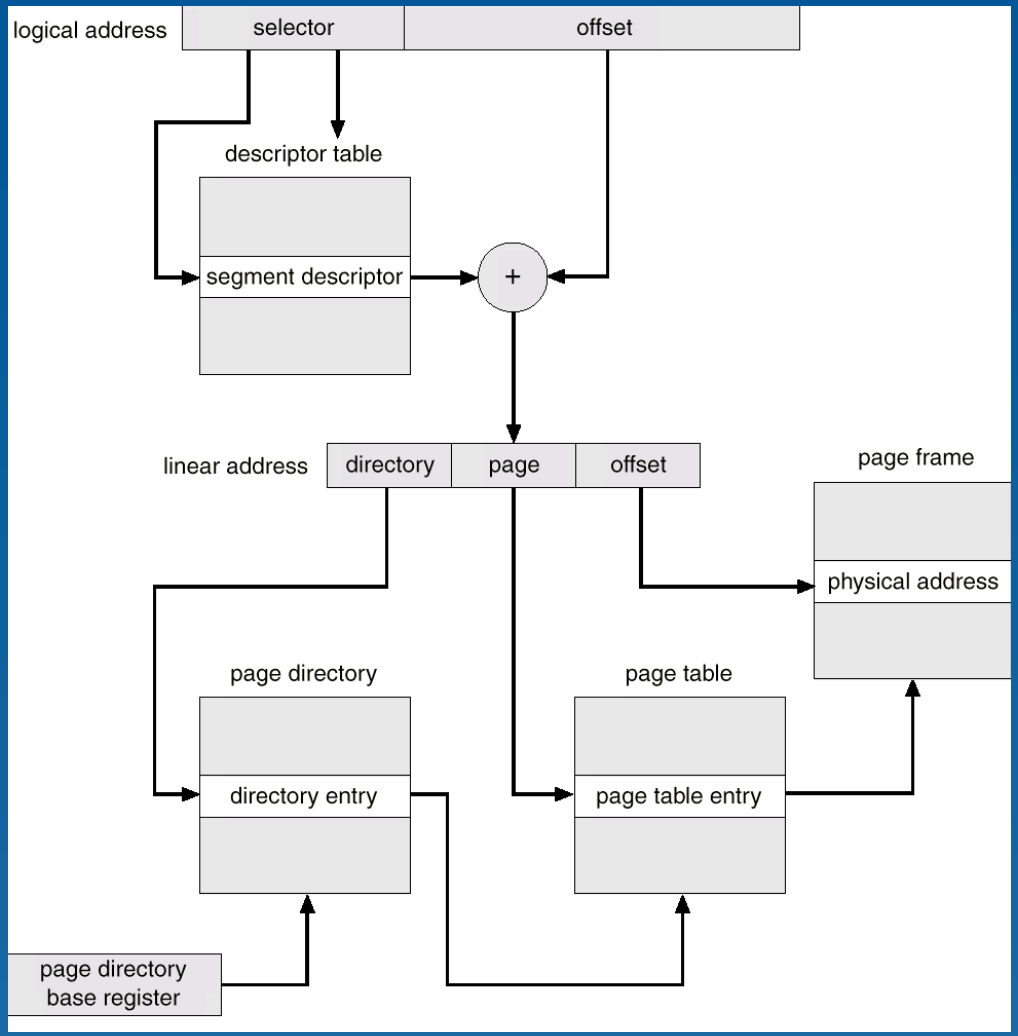
i386 的段式管理


logical address
selector
offset
descriptor table
segment descriptor
+
32-bit linear address

i386 的页式管理


(logical address)
page directory
page table
offset
31
22 21
12 11
0
page table
4-KB page
page directory
CR3 register
4-MB page
page directory
offset
31
22 21
0

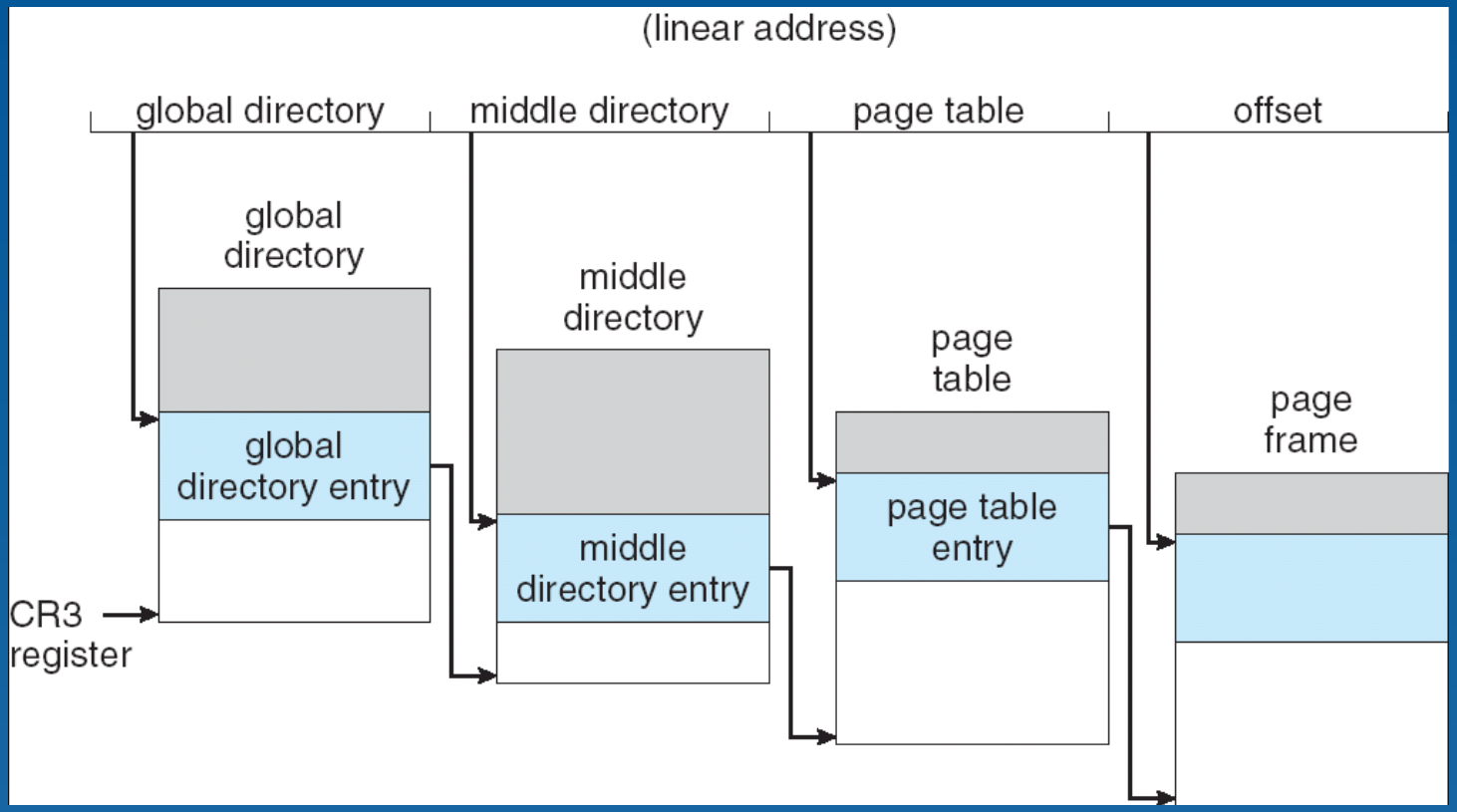
Intel i386 完整的地址翻译



Linux 的逻辑地址

把逻辑地址划分成 4 部分

Linux 的三级页表


(linear address)
global directory
middle directory
page table
offset
global directory
global directory entry
middle directory
middle directory entry
page table
page table entry
page frame
CR3 register


End