计算机组成原理

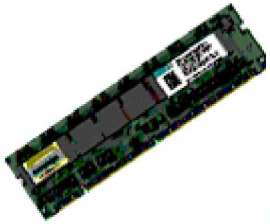
COD第5章 虚拟存储器

llxx@ustc.edu.cn

本章内容

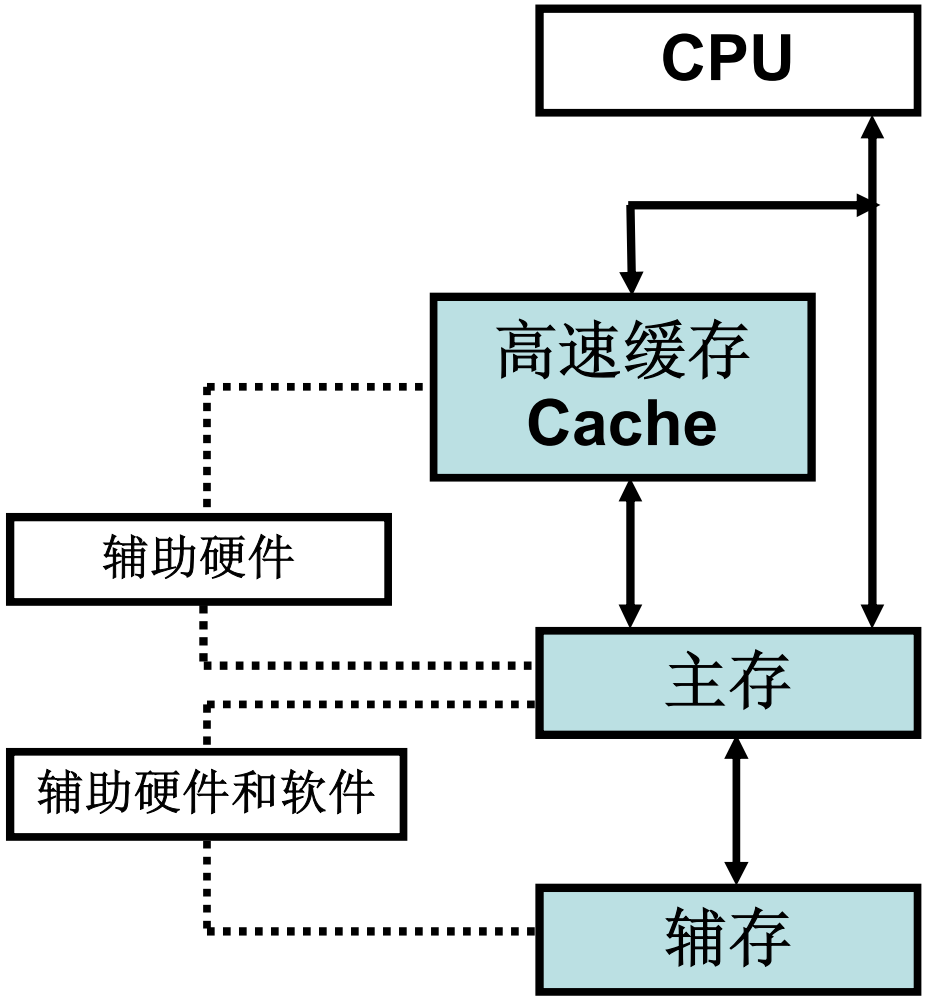
- ✓5.4 虚拟存储器
  - ✓实方式 vs 虚方式
  - ✓虚存管理机制
  - ✓TLB
  - ✓MMU
  - ✓层次化：Cache-Memory-Disk



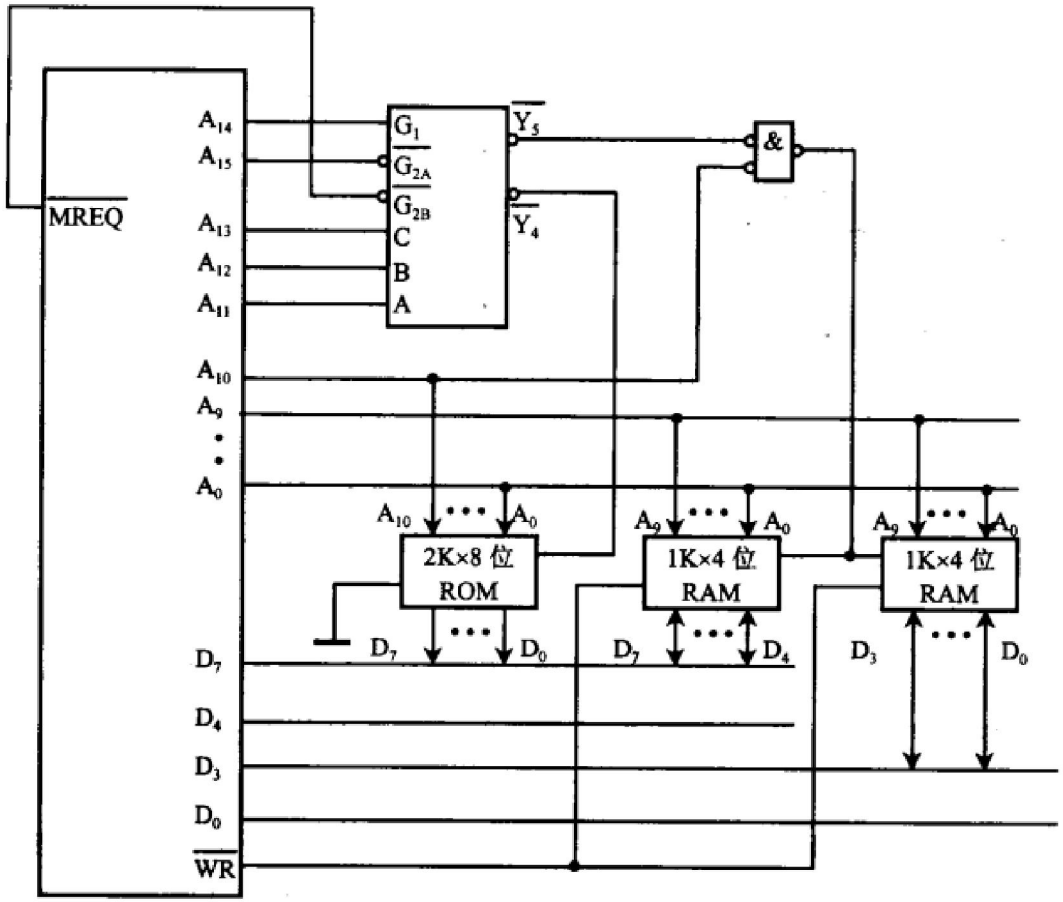


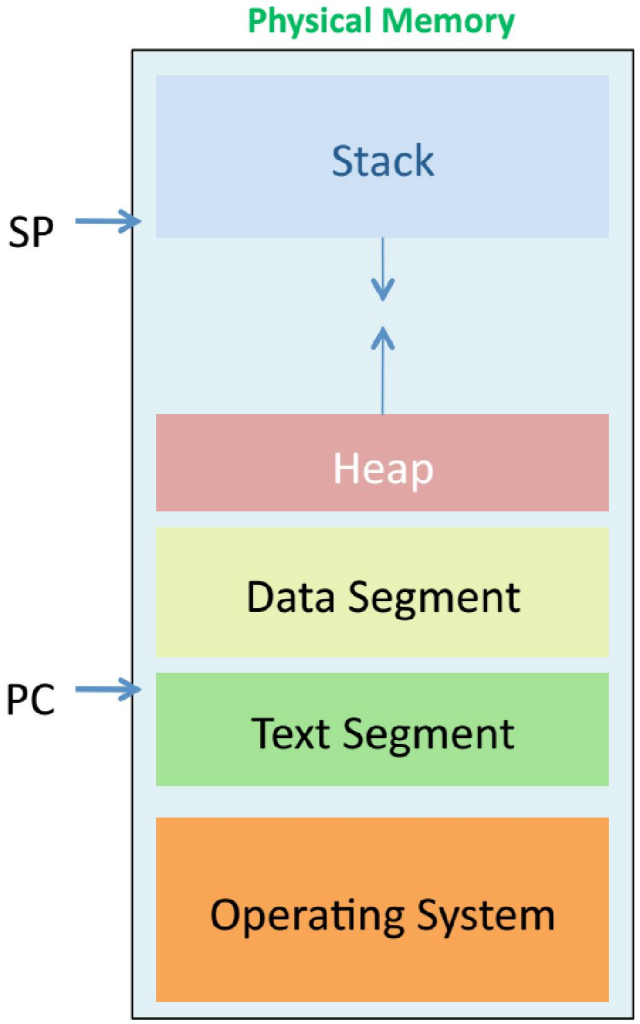

三级存储体系结构

- 三级存储系统：
  - 缓存
  - 主存
  - 辅存
- 缓存－主存层次
  - 映射、查找、读写，替换
- 主存－辅存层次
  - 映射、查找、读写、替换



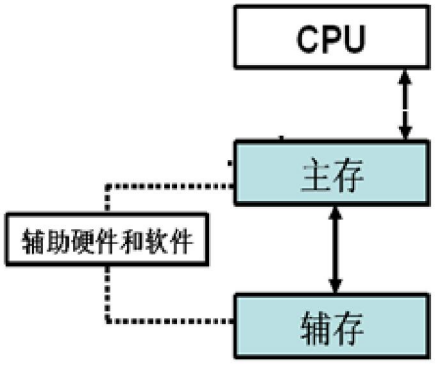
实模式访存：单任务，物理存储器





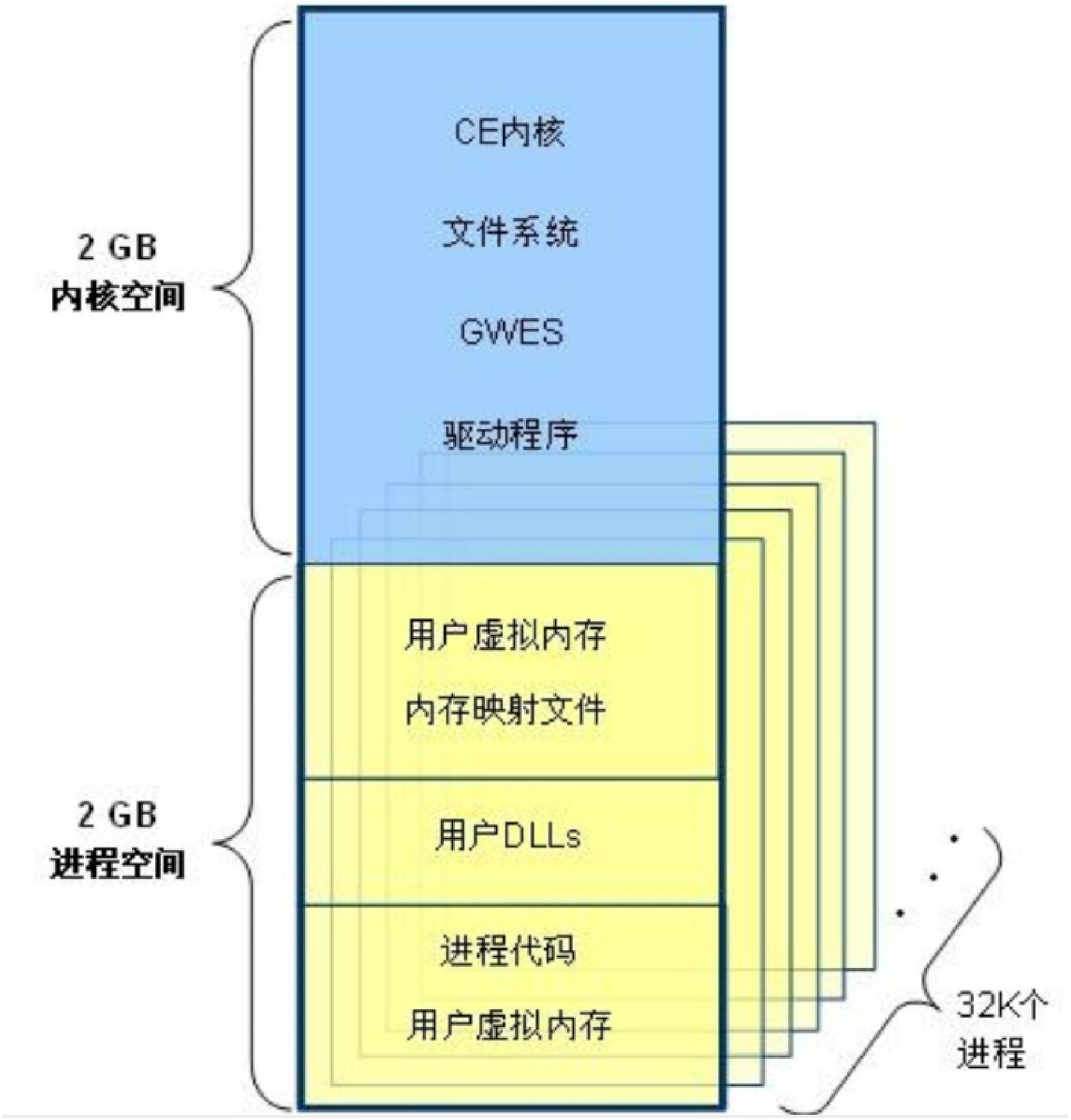
虚拟存储系统（Virtual memory）

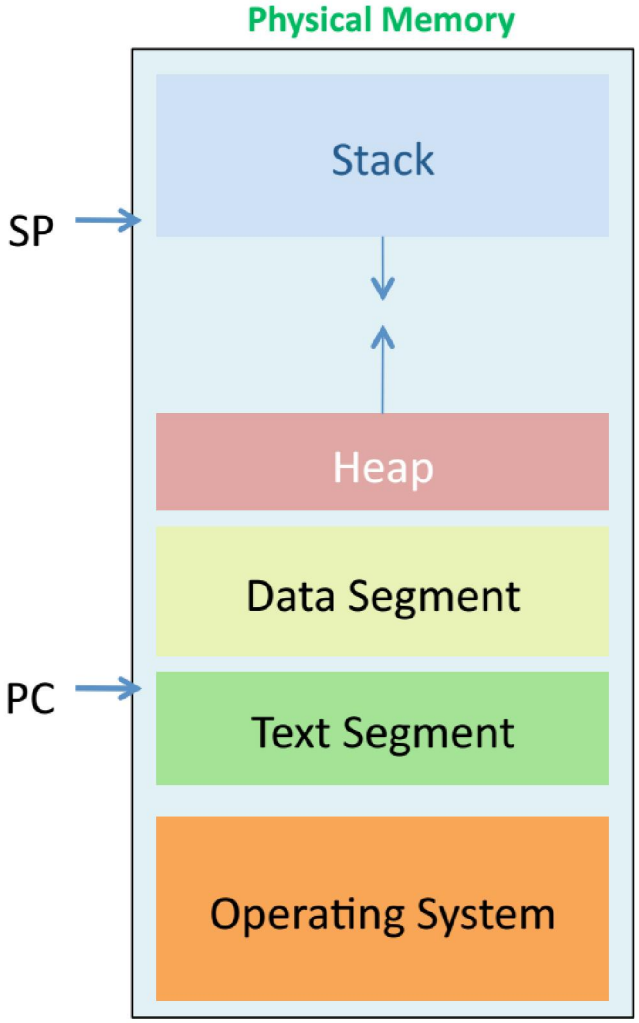
- 1961年曼彻斯特大学提出
  - 程序要求的存储器空间越来越大
    - 虚存 = 主存 + 辅存
      - 例：CPU地址总线64位，主存4G，磁盘100G
  - 多任务间的代码和数据访问保护



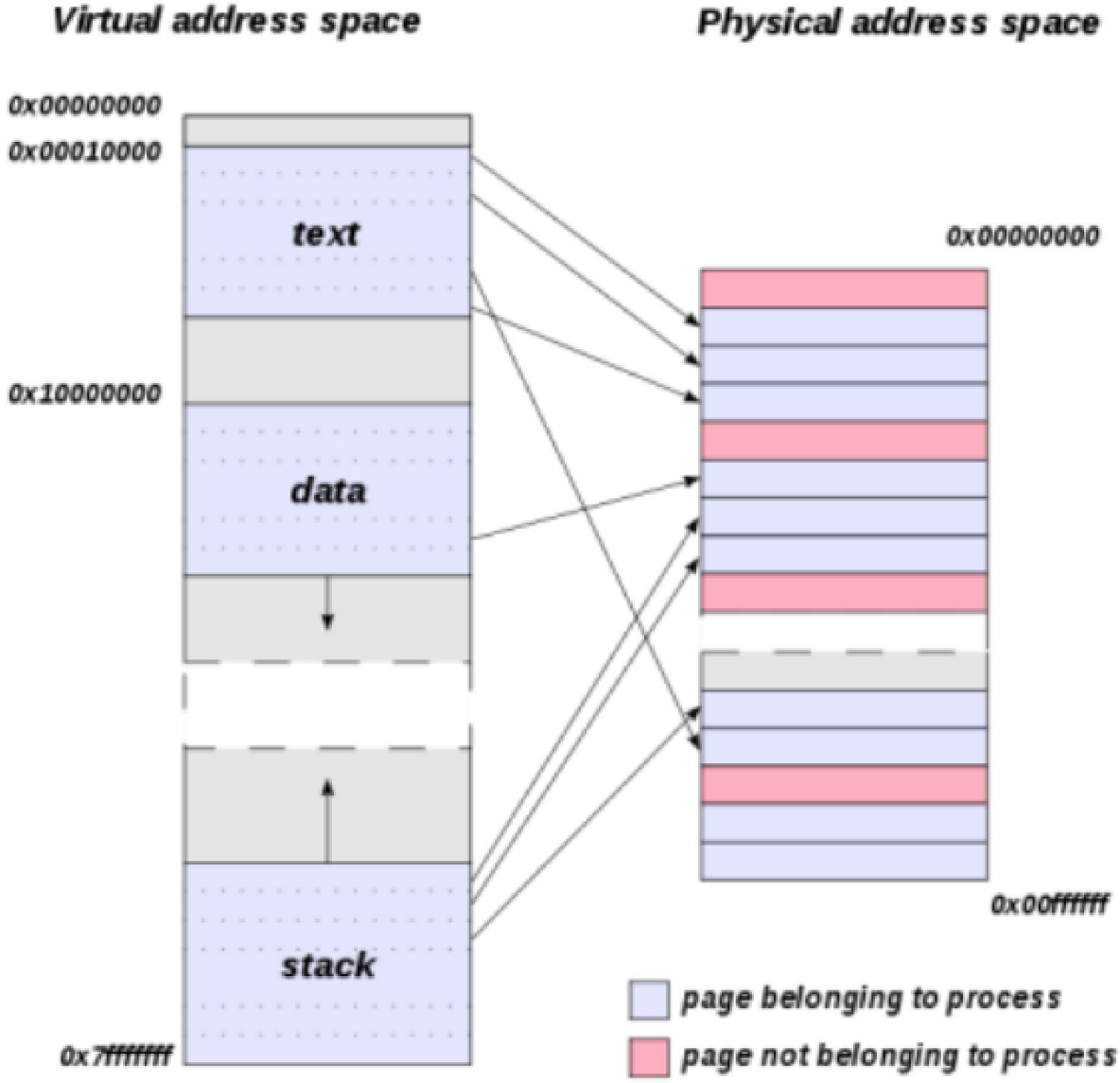
- 现代虚拟存储系统：将主存作为外存的缓存
  - 虚存由硬件（MMU）和OS存储管理器共同管理
    - 使对辅存的访问速度接近主存的速度
  - 不同应用具有不同的地址空间和访问权限
    - 程序的地址空间是以“0”地址起始的逻辑地址集合
    - 这个空间中的地址都是相对地址，也称为逻辑地址。

多任务系统虚存空间划分示例-wince





程序逻辑地址空间与内存物理地址空间

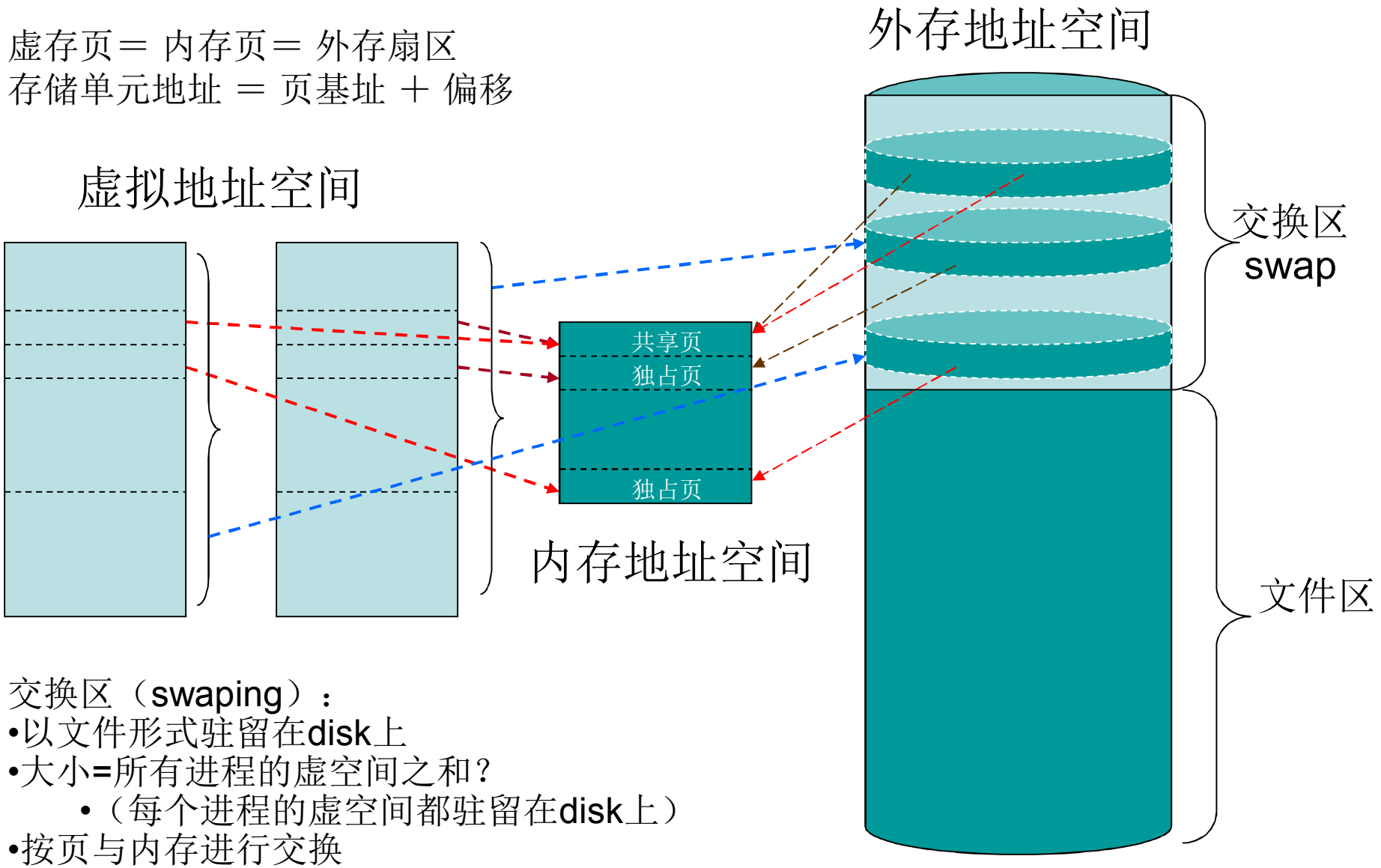


虚拟存储器技术的关键问题

- 访存过程：**映射、查找、读写、替换**
  - 程序的虚空间存在于辅存中，主存只是其镜像，称“页框”
- 映射：虚拟存储器地址空间管理方式
  - 页式管理：虚存和主存都划分成固定大小的页
  - 段式管理：段为程序的逻辑单位
    - 段表结构： 段名、地址、装入位、段长、访问方式
    - 段表基址寄存器：指明段表的启始地址。
  - 段页式管理
- 虚存管理实现机制：由MMU+OS实现，对应用透明
  - 查找TTW：基于表进行地址变换和查找
    - 保证不同进程的虚拟地址不会映射到相同的物理地址上
    - 地址变换速度：TLB(Translation Look-Aside Buffers)
  - Page Fault：页面失效，缺页
    - 决定把作业的虚拟地址空间的哪一部分装入主存，以及放在主存的什么位置；
  - 替换：主存空间不够时把哪一部分置换出主存

例：页式虚存，内存，外存

虚存页＝ 内存页＝ 外存扇区
存储单元地址 ＝ 页基址 ＋ 偏移



交换区（swaping）：
- 以文件形式驻留在disk上
- 大小=所有进程的虚空间之和？
  - （每个进程的虚空间都驻留在disk上）
- 按页与内存进行交换

Address Translation：TTW

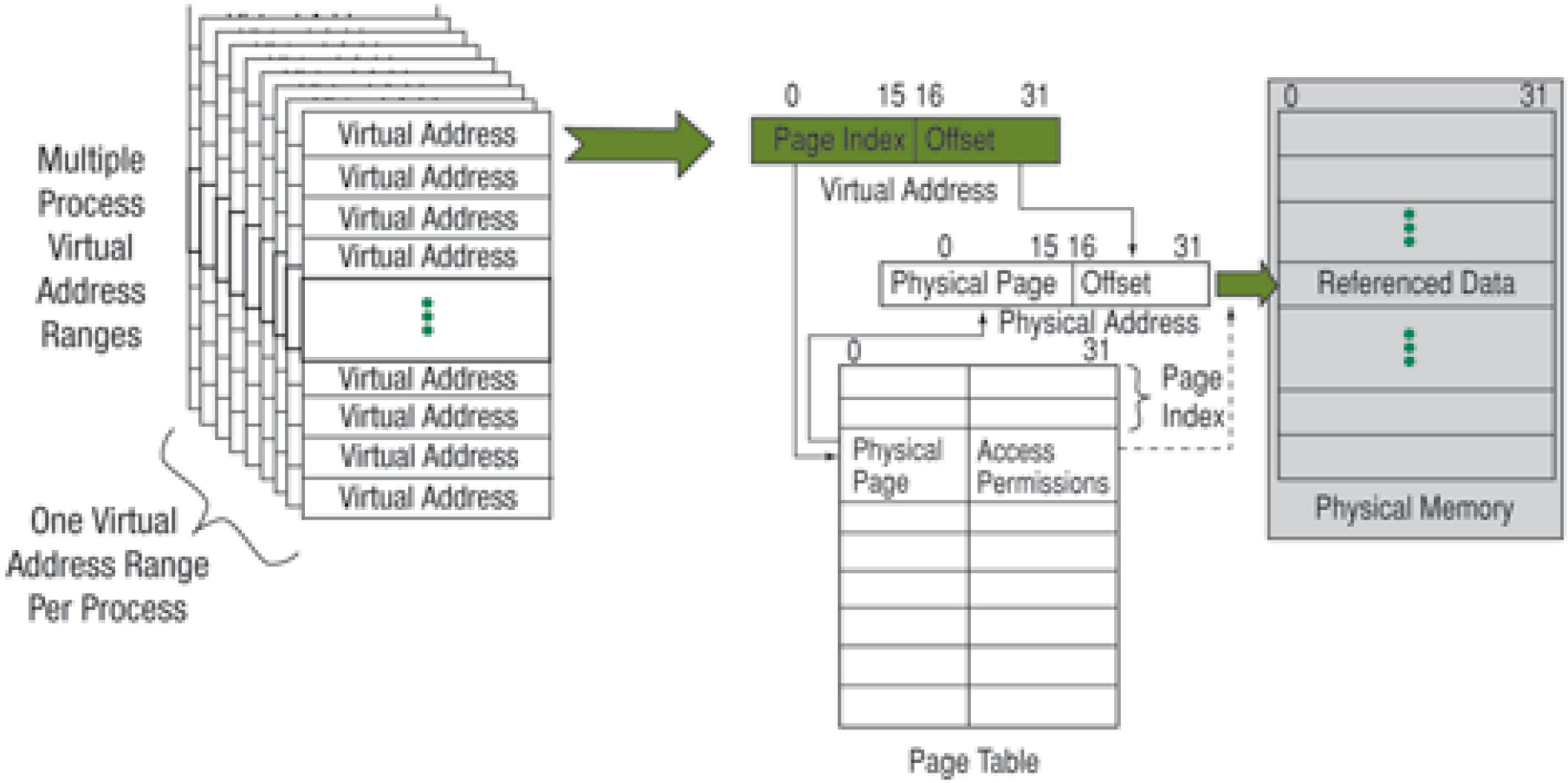
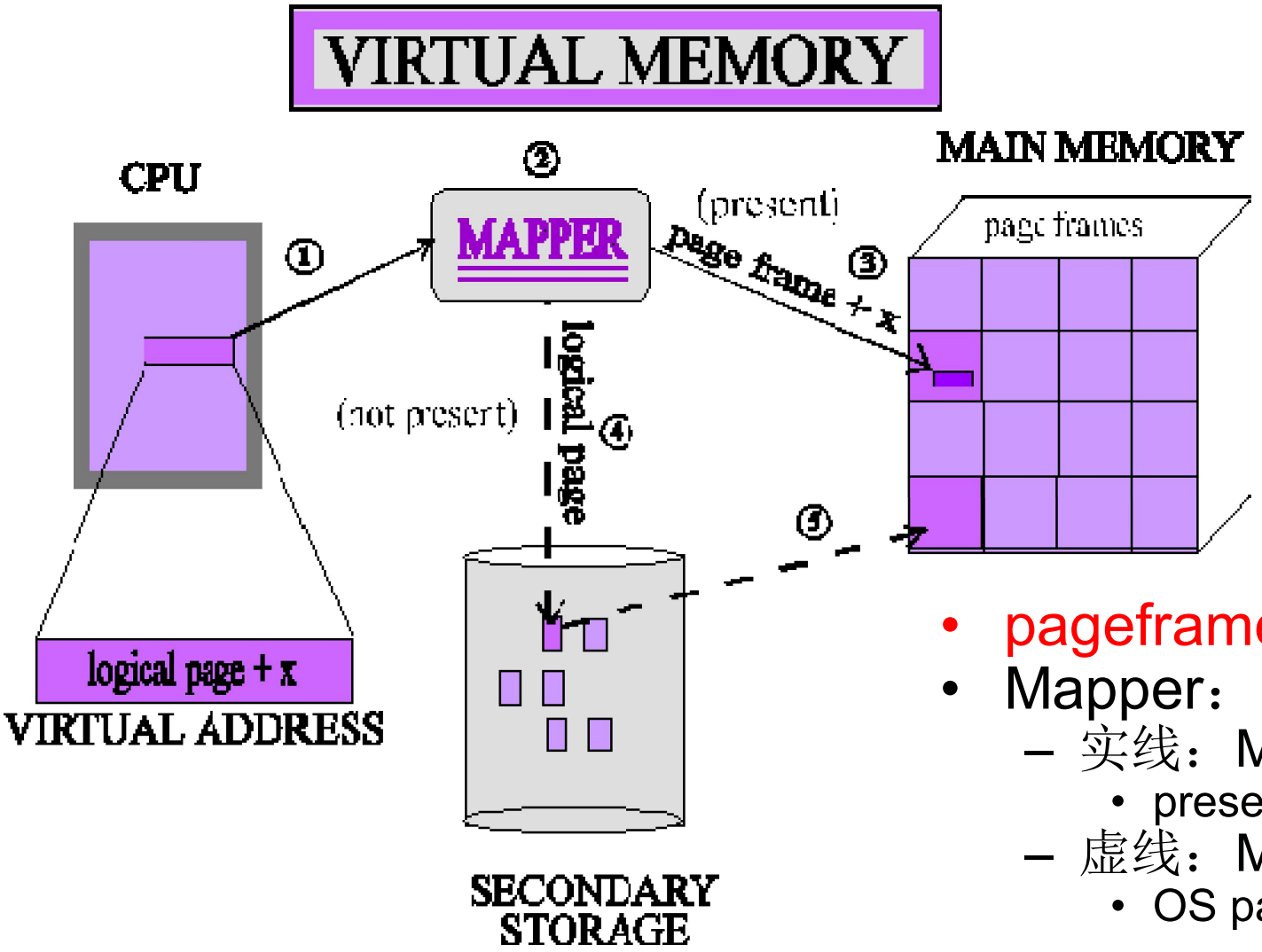


Figure 2 In a virtual memory system, addresses in each user's virtual address space are translated to reference code and data in physical memory.

CPU的访存操作：Mapper



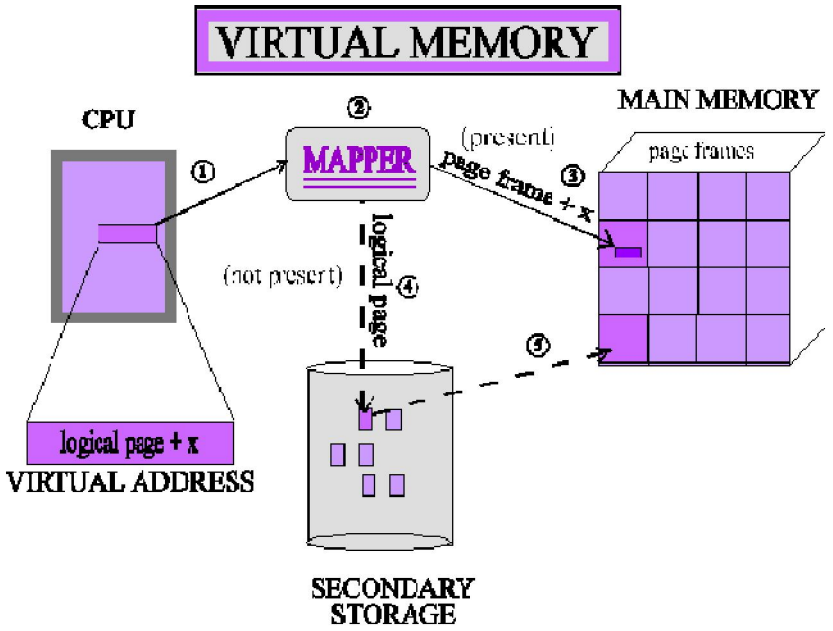
- pageframe
- Mapper：MMU＋OS
  - 实线：MMU
    - present = valid
  - 虚线：MMU发出中断
    - OS pagefault

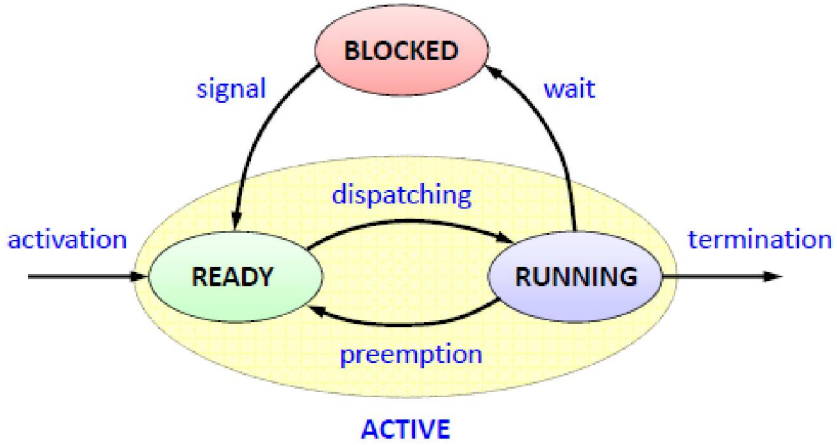
访存流程图：（无TLB和Cache）

1.CPU给出虚地址

2.Mapper(查慢表)

- 如果Entry存在
  - 如果Valid，则访问
  - 如果Invalid，转OS缺页异常
    - 原进程进入中断态？
    - page fault处理：调Disk页
      - » 写回、替换？
      - » 读入请求页：启动DMA？
      - » *切换执行新进程？*
      - » DMA中断：请求页已读入
      - » *OS切换回原进程*
    - 重试原进程产生缺页的指令
- 如果Entry不存在
  - 原因？转“错误异常”



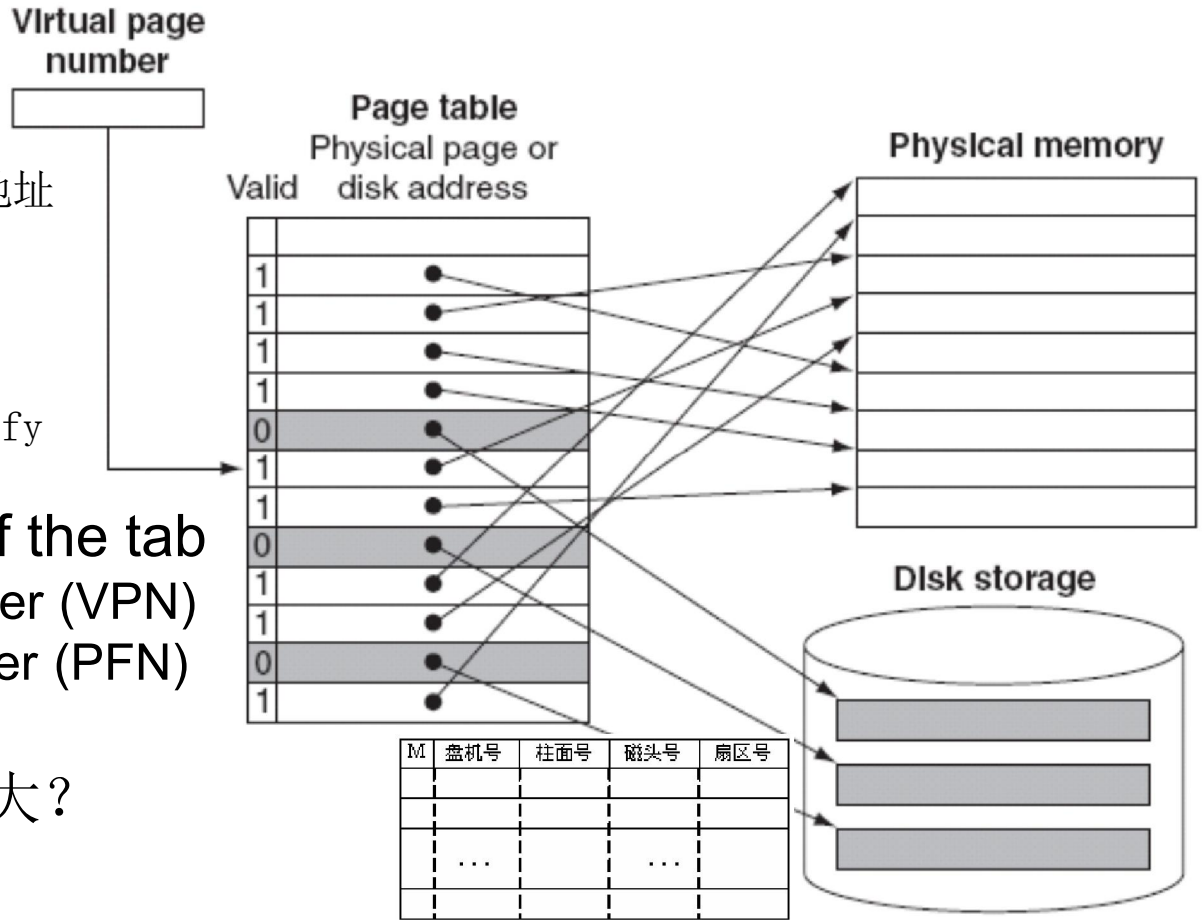


页表：虚实映射

- 实现虚地址与内存地址或磁盘地址的映射
- OS创建：每个进程一个，存放于内存中
  - 保证每个进程的不同虚拟地址不会映射到相同的物理地址！
  - 同时运行的多个进程的物理内存尽量不要相互重叠！

- 控制位
  - Valid：装入
    - 未装入页：disk地址
  - Dirty
  - Count/Ref
  - Access Rights
    - read/write/modify
    - 作用：页保护
- VPN is the index of the tab
  - virtual page number (VPN)
  - page frame number (PFN)

- 页表含多少项？多大？



- 每页size=4K。
- 虚空间16页（64k）
- 系统物理内存4页（16K）

- 页表
- 16个entry，每项对应逻辑地址空间的一个页
- Frame域表明了该逻辑页存储在哪一个物理页框中

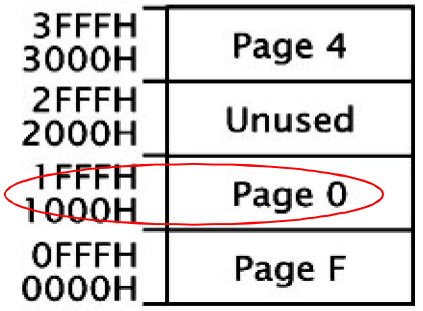
	Frame	Valid	Count	Dirty
0	0 1	1	1 0	0
1	0 0	0	0 0	0
2	0 0	0	0 0	0
3	0 0	0	0 0	0
4	1 1	1	0 1	1
5	0 0	0	0 0	0
6	0 0	0	0 0	0
7	0 0	0	0 0	0
8	0 0	0	0 0	0
9	0 0	0	0 0	0
A	0 0	0	0 0	0
B	0 0	0	0 0	0
C	0 0	0	0 0	0
D	0 0	0	0 0	0
E	0 0	0	0 0	0
F	0 0	1	0 0	0

(a)

（a）为页表
（b）为对应的物理内存

物理页框
•3
•2
•1
•0

装入的虚存页号



(b)

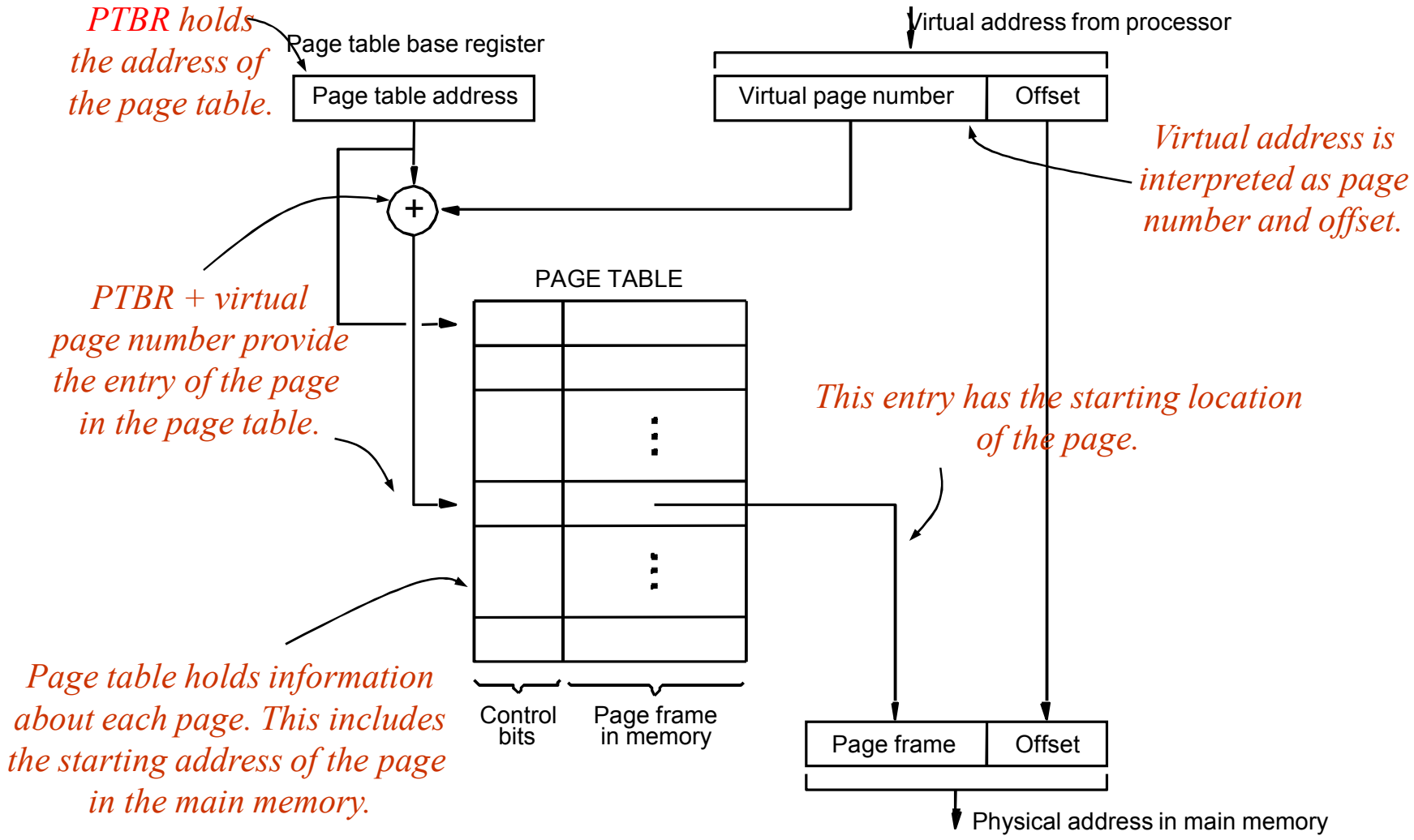
本例，虚存的第0页：
逻辑地址0000H~0FFFH
存储在内存的1号页框中，物理地址为1000H~1FFFH

例：虚拟地址＝0xxxH

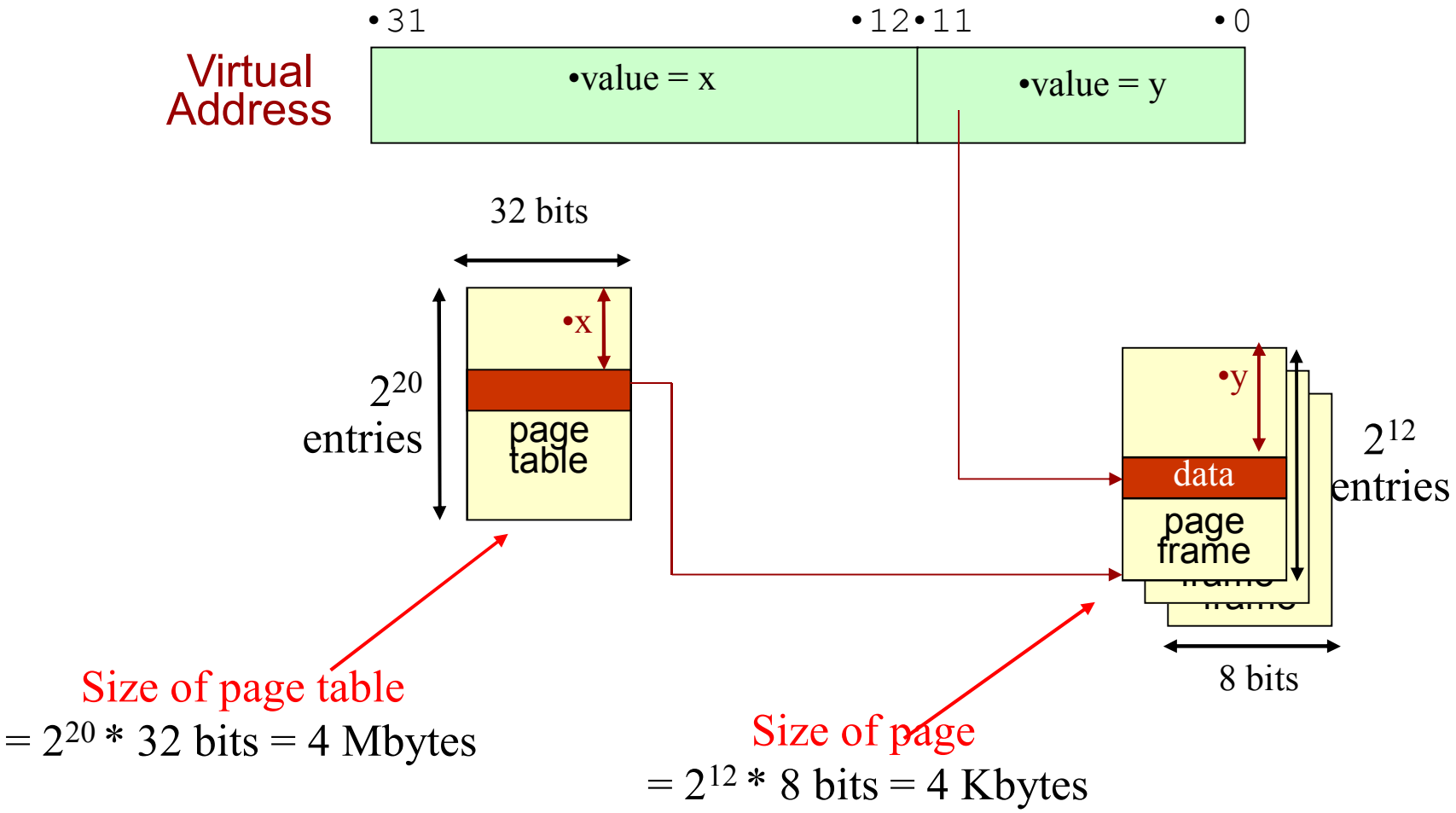
虚页号　页内偏移

•如果访问虚存page “A” ？
•执行一条访存指令需访问几次存储器？

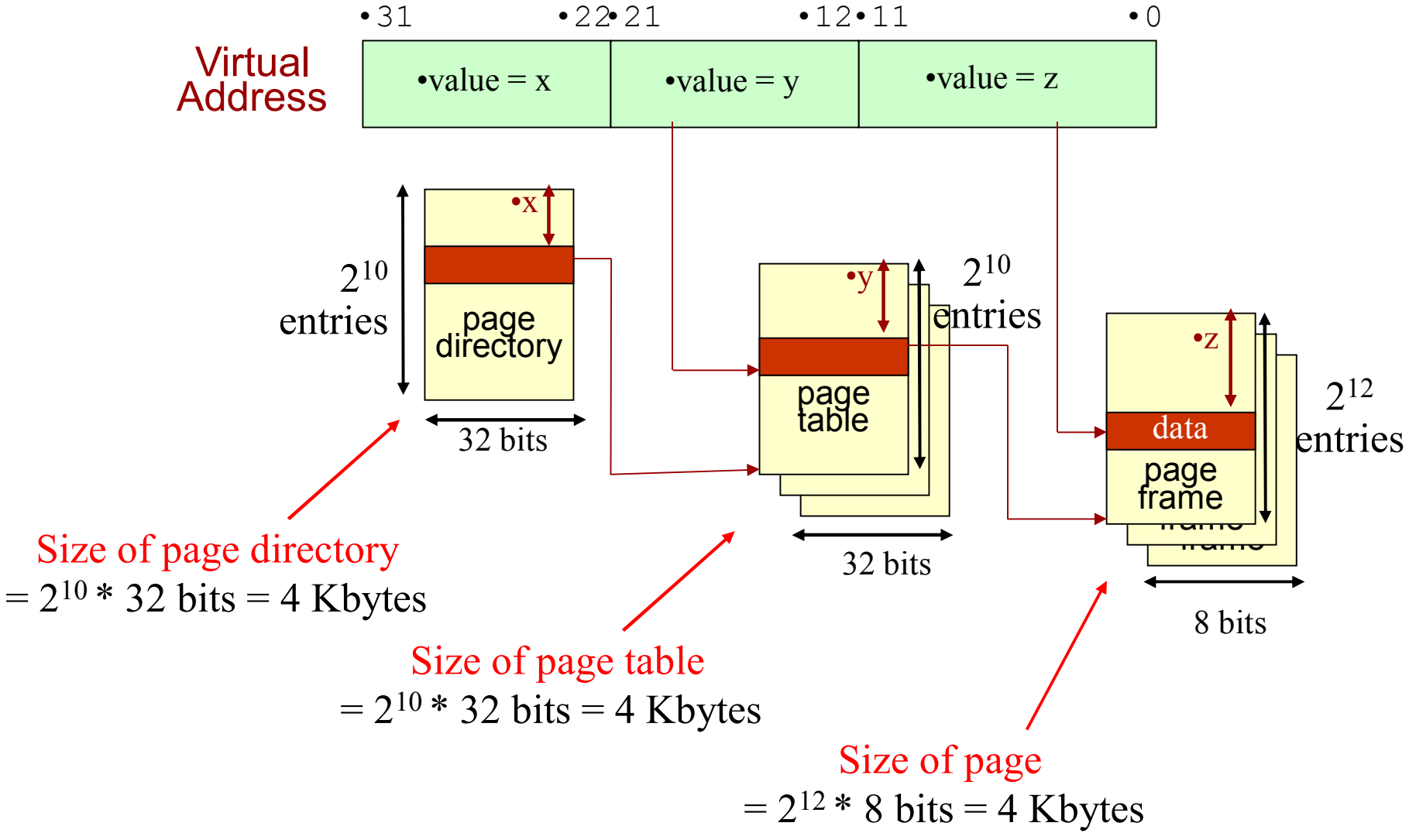
PTBR：页表在哪儿？


PTBR holds the address of the page table.
Page table base register
Page table address
Virtual address from processor
Virtual page number
Offset
Virtual address is interpreted as page number and offset.
PTBR + virtual page number provide the entry of the page in the page table.
PAGE TABLE
This entry has the starting location of the page.
Page table holds information about each page. This includes the starting address of the page in the main memory.
Control bits
Page frame in memory
Page frame
Offset
Physical address in main memory

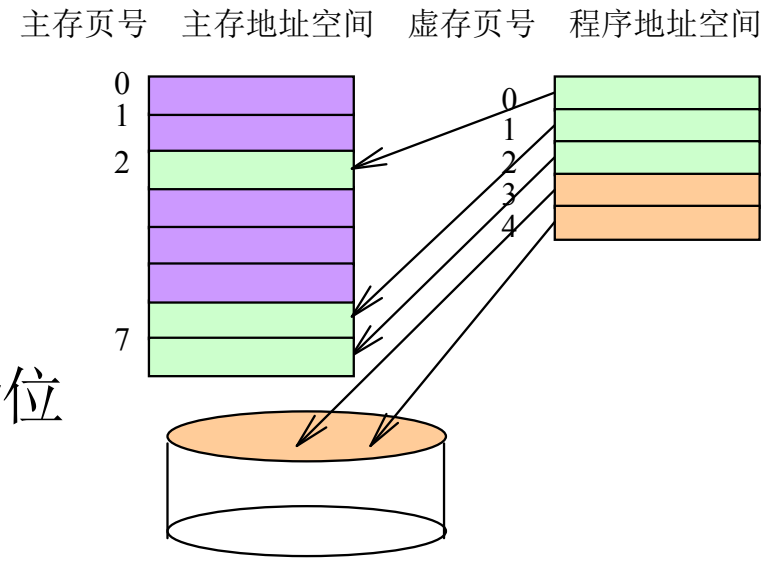
Single-Level Page Table：一级太大！



Twolevel Page Table

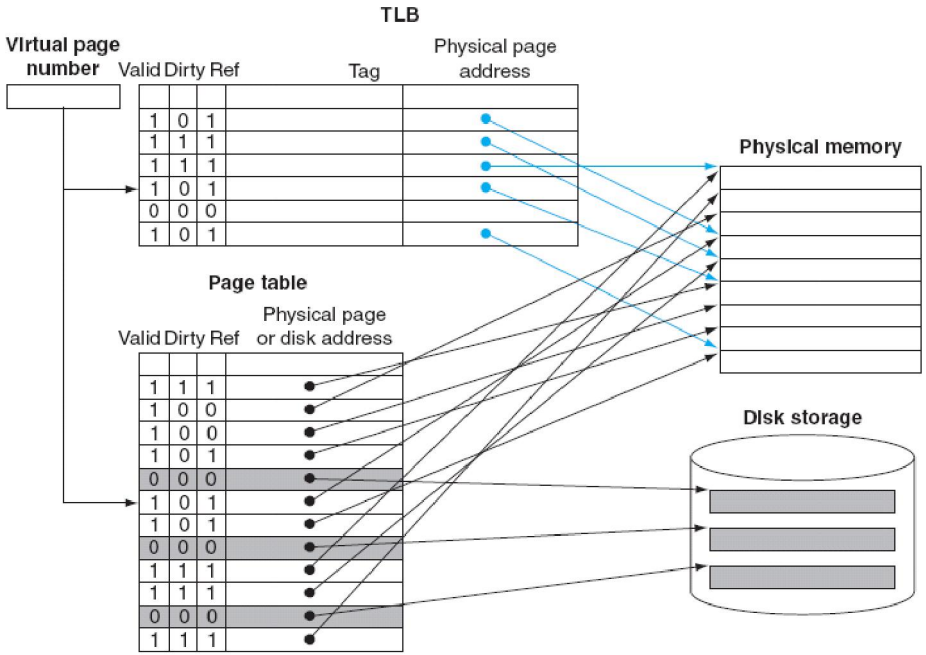


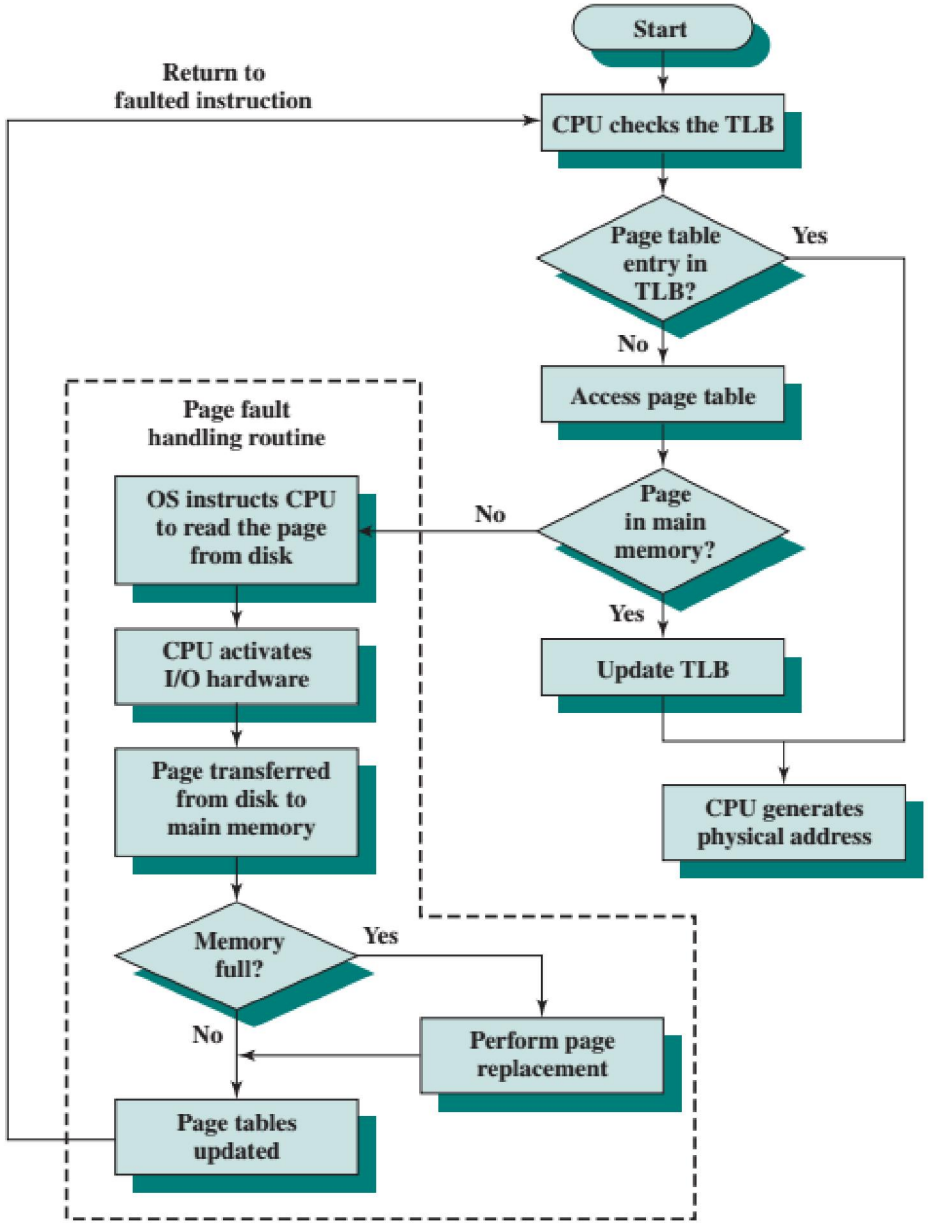
页式虚存管理实现中的问题



- 虚存与物理内存映象规则
  - 全相联（冲突小）
- 写策略
  - 写回法（copy－back）：Dirty位
- 替换算法：
  - FIFO，LRU（Count位），NUR（最近未使用，Ref位）
- 异常：缺页（页面失效），非法访问（权限）
  - 在一条指令的执行过程中发生
  - 切换到其它已就绪的任务(进程)
    - 采用后援寄存器技术、预判技术
- 提高查表速度：慢表（page table）/快表（TLB）

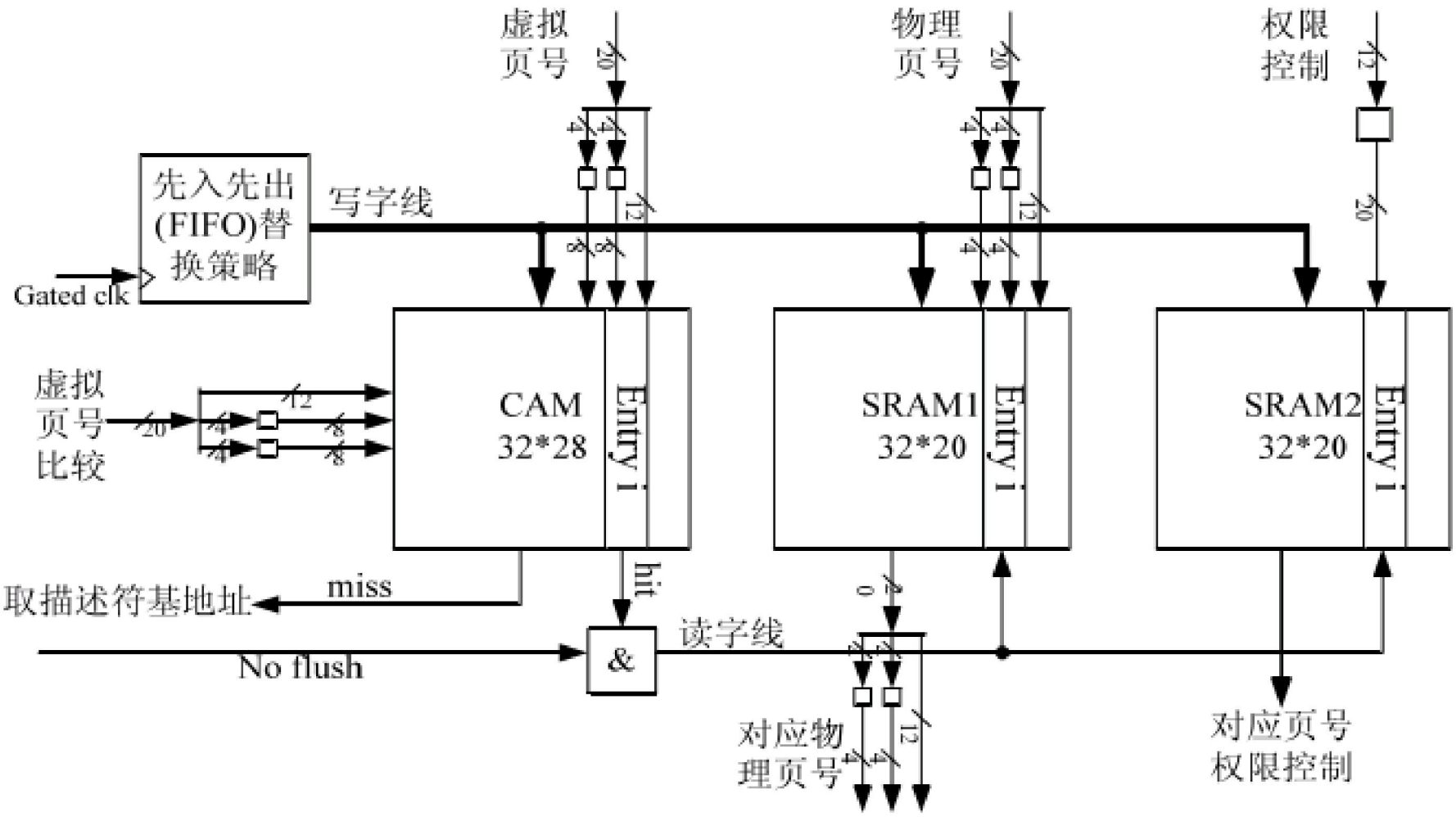
快表TLB(Translation Look-aside Buffers)





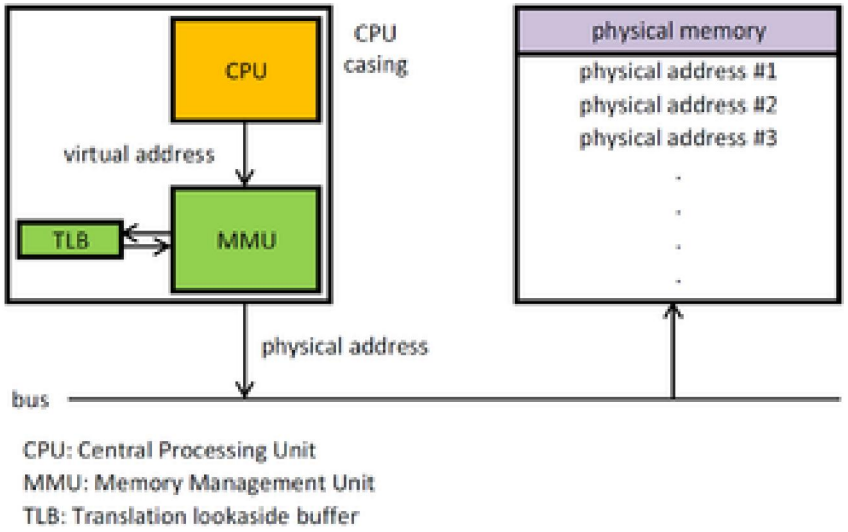
- 映射方式：fully associative, **set associative**, direct mapped
- Tag?
- 替换策略：随机，FIFO
- Operation of Paging and TLB

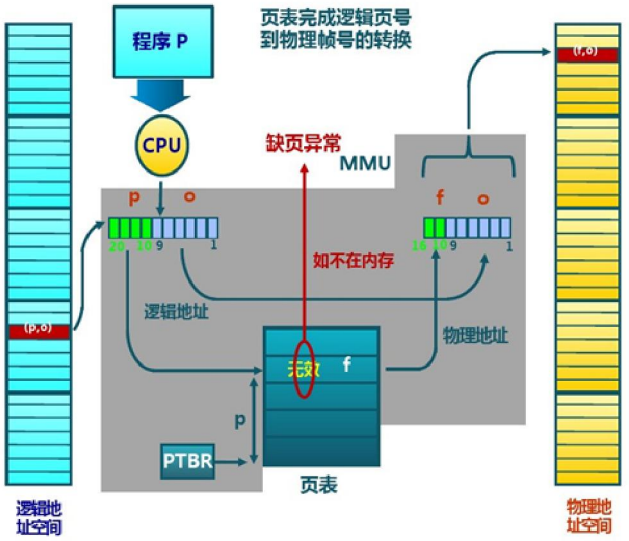
TLB实现


虚拟
页号
物理
页号
权限
控制
先入先出
(FIFO)替
换策略
写字线
Gated clk
虚拟
页号
比较
CAM
32*28
Entry i
SRAM1
32*20
Entry i
SRAM2
32*20
Entry i
取描述符基地址
miss
hit
&
读字线
No flush
对应物
理页号
对应页号
权限控制

MMU（Memory Management Unit）

- 主要功能
  - 地址转换：虚地址映射成物理地址。
    - MMU关掉时，虚地址直接输出到物理地址总线
  - 访问控制：检查指令对当前页的访问权限
  - MMU fault异常请求
    - 缺页，转换错，非法访问，数据对齐错
    - MPIS CPU：CP0 Bad_Address Register记录发生异常的地址
- 组成：TLB，PTBR，AccessPermisionControl

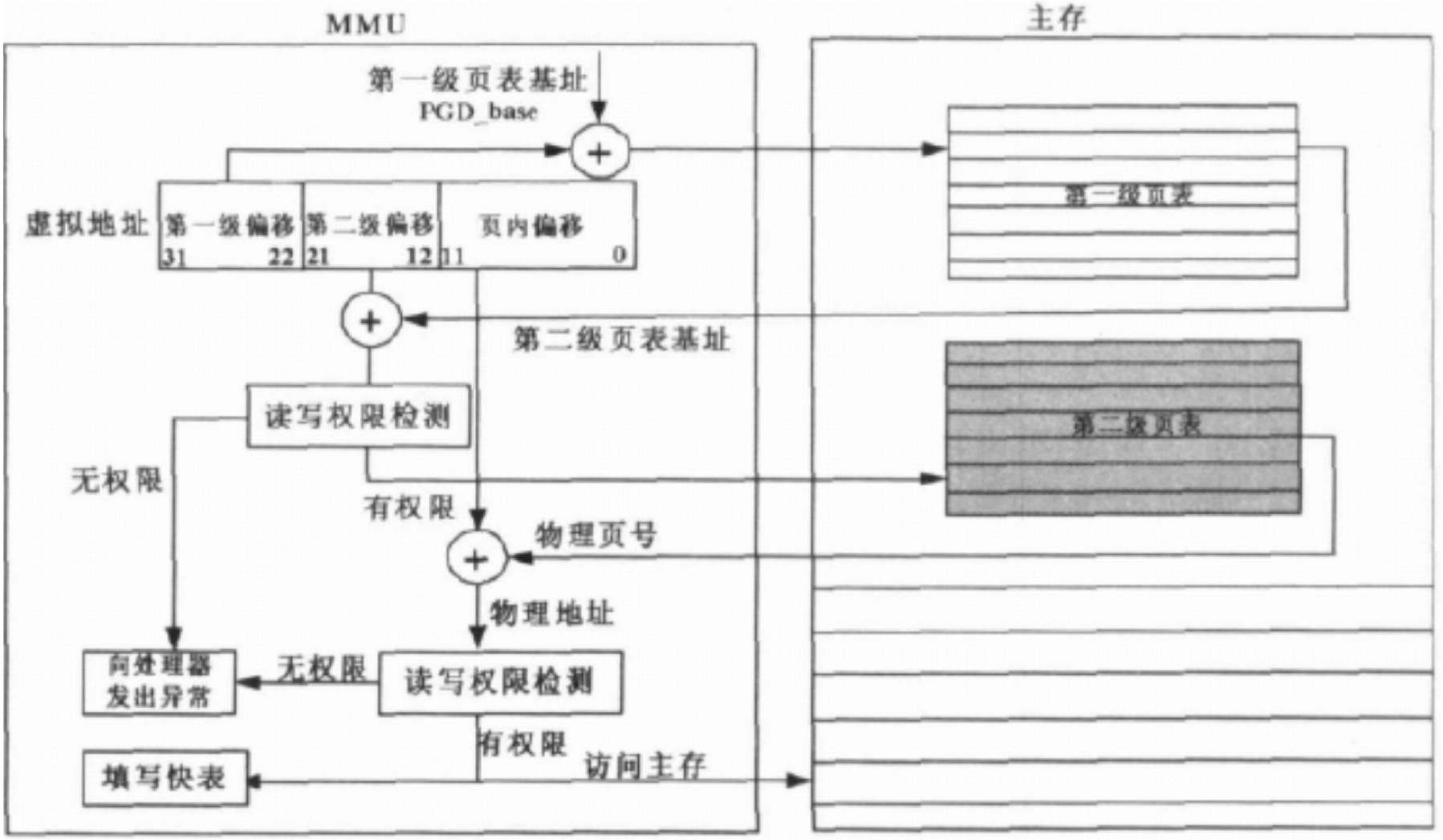




MMU工作过程：MMU使能

- 地址空间重定位：在或激活一个进程时
  - OS将该进程的页表基址填入CPU页表基址寄存器(PTBR)
  - 调用TLB flush指令，将TLB数据置为无效。
    - 频繁切换？MMU可以锁定某些TLB项，以提高特定地址变换速度
- 虚实地址映射：查TLB
- TTW：TLB miss
  - 软件法：MMU 发出异常请求后, 由OS查询页表并填充TLB
    - 如MIPS 4Ke
  - 硬件法：MMU 中设立专门的硬件完成页表查询和TLB填充
    - CISC CPU大多使用这种方案, 如x86 CPU;
    - 如ARM11 MPCore Processor
- 发出MMU异常
  - 缺页，转换错，非法访问，数据对齐错

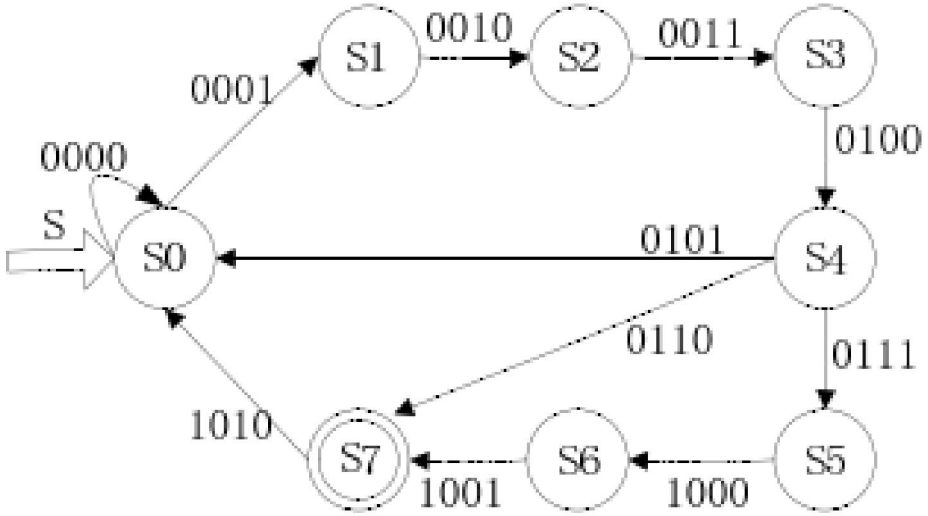
TLB miss时MMU的TTW（硬件）



流水线停顿？（阻塞、非阻塞）

MMU控制器设计例：TTW过程

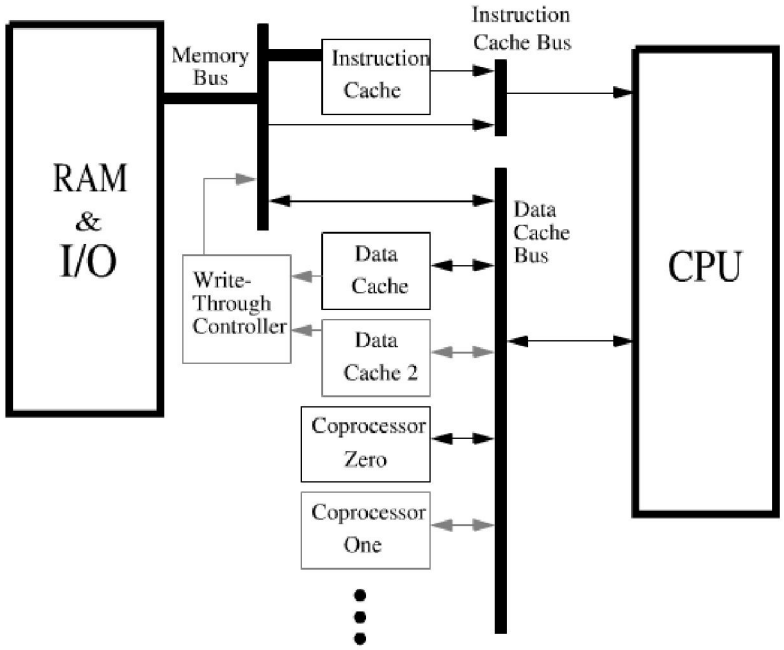
状态标号	功能描述
S0	MMU 初始化状态
S1	获取 C2 寄存器和 mva 中的指定位
S2	计算访问第一级描述符的物理地址
S3	在外部存储器 RAM 中获得第一级描述符
S4	进行第一级描述符类型的判断
S5	计算访问第二级描述符的物理地址
S6	在外部存储器 RAM 中获得第二级描述符
S7	计算出转换虚拟地址 mva 对应的物理地址



•应包括：TTW，异常等过程

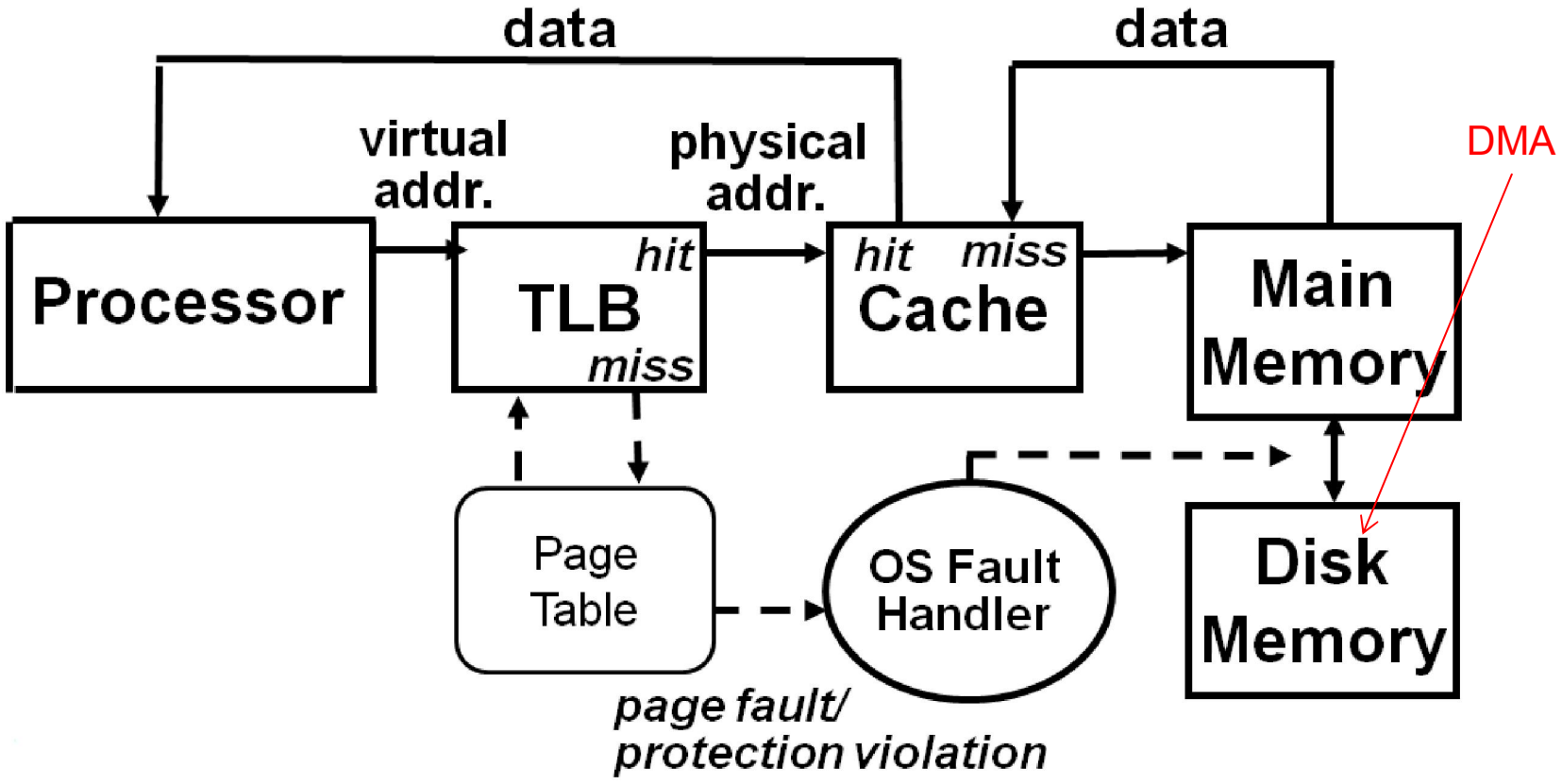
CP0

- 含32个寄存器，控制CPU的工作模式
  - Register 0, Index
    - 作为MMU的索引用
  - Register 4, Context
    - 用以加速TLB Miss异常的处理
  - Register 8, BadVAddr
    - 发生TLB Miss或Address Error时的地址
  - Register 9, Count
    - 计数频率是系统主频的1/2
  - Register 11, Compare
    - Compare和Count相等则触发硬中断IP7
  - Register 12, Status
    - 特权级，中断屏蔽，中断允许等。
  - Register 13, Cause
    - 异常的原因(interrupt pending)。
  - Register 14, EPC
    - 异常发生时系统正在执行的指令的地址
  - Register 18/19, WatchLo/WatchHi
    - 用于设置硬件数据断点
  - Register 28/29, TagLo和TagHi
    - 用于高速缓存(Cache)管理。



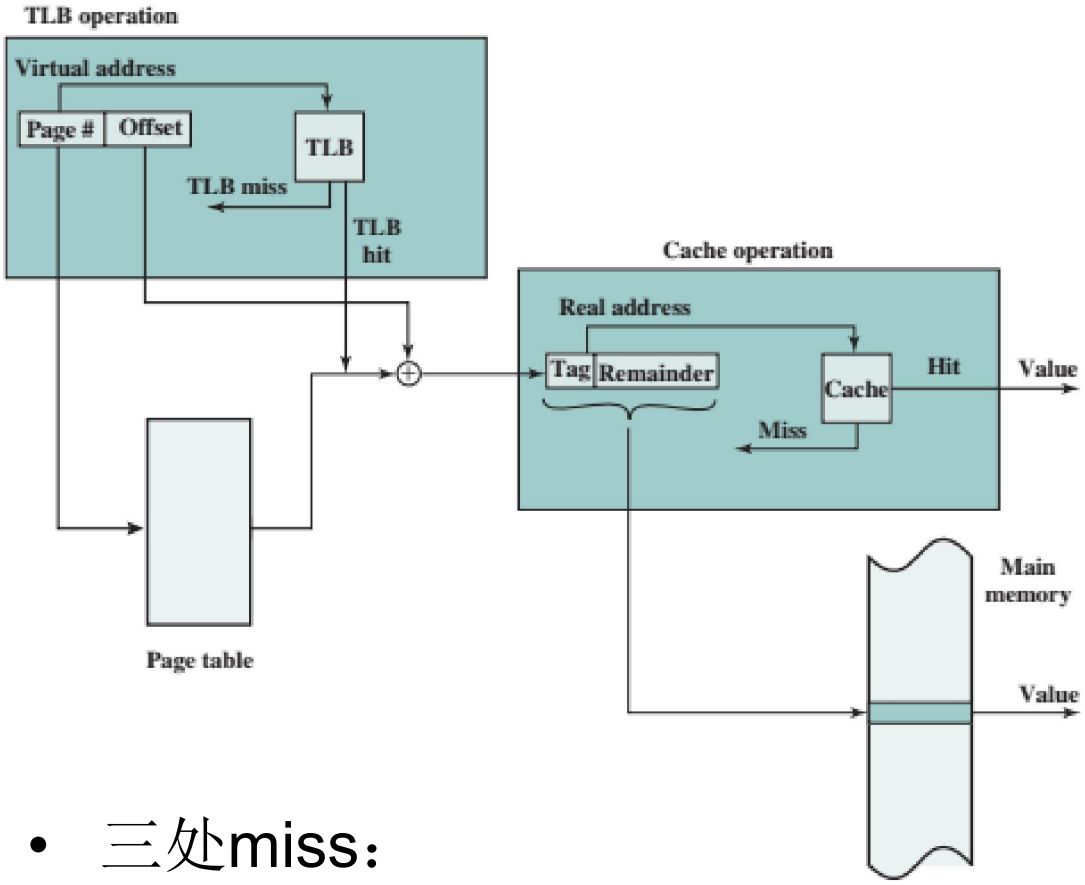
MMU，Cache，OS

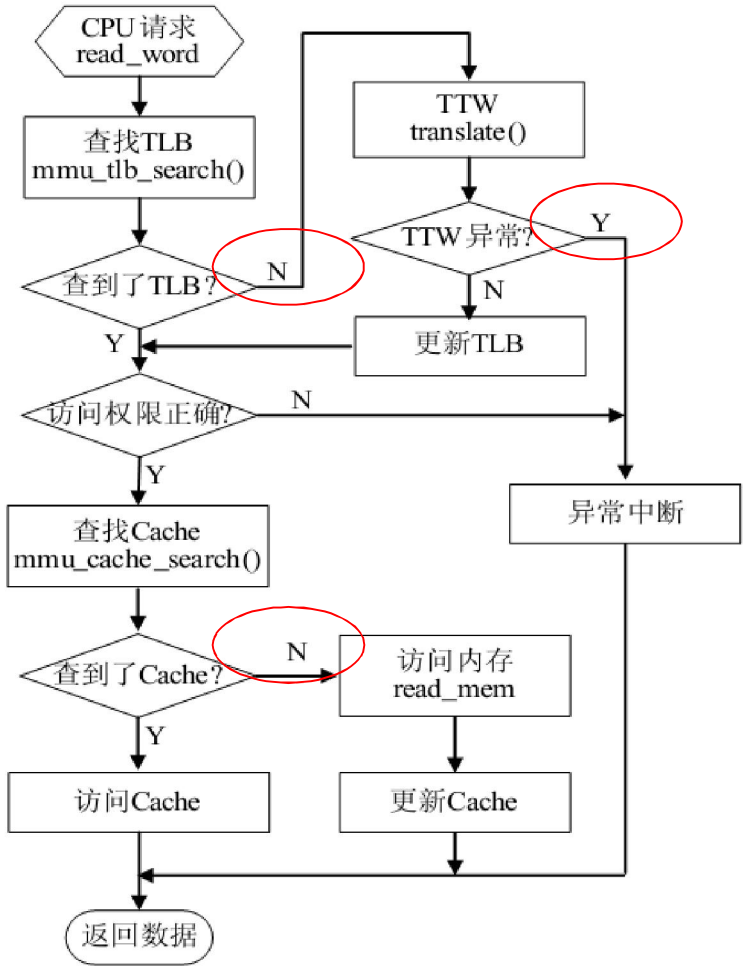
- 物理地址Cache



- 虚地址Cache：

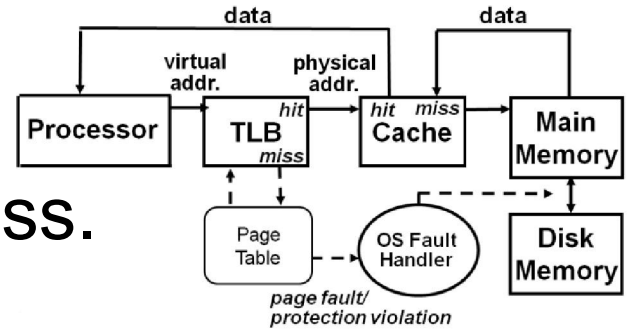
访存流程图：MMU，Cache，OS





- 三处miss：
  - TLB miss：CPU stall？
    - 硬件TTW，软件TTW
  - Page fault：进程切换？
  - Cache miss：CPU stall？

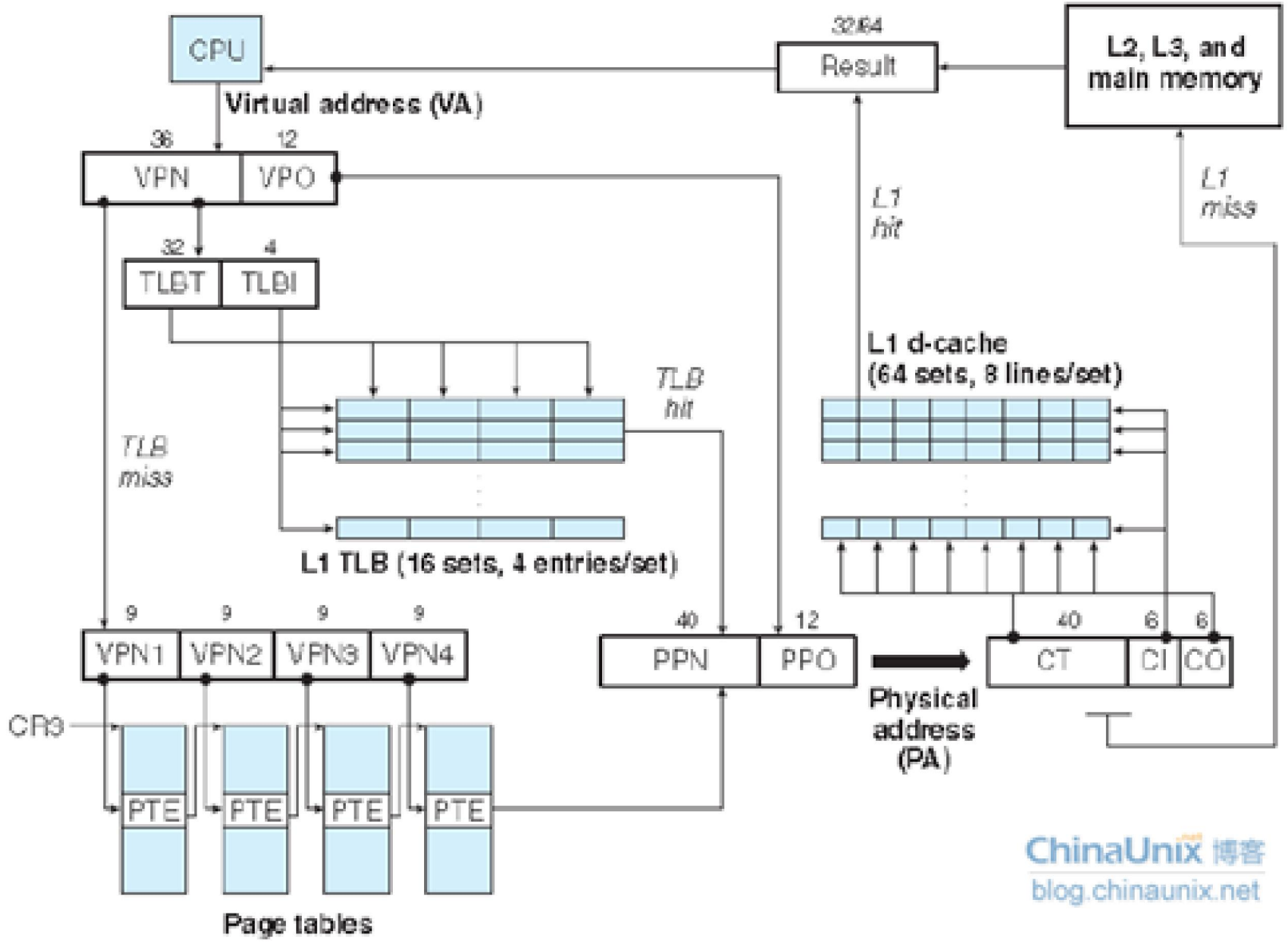
访存异常（见COD5中文版第301页）



- 一次访存可能有三种miss
  - TLB miss, page fault, cache miss.
- 事件组合：物理Cache
  - 第一种可能但不会被发现
  - 三种可能：后续处理？

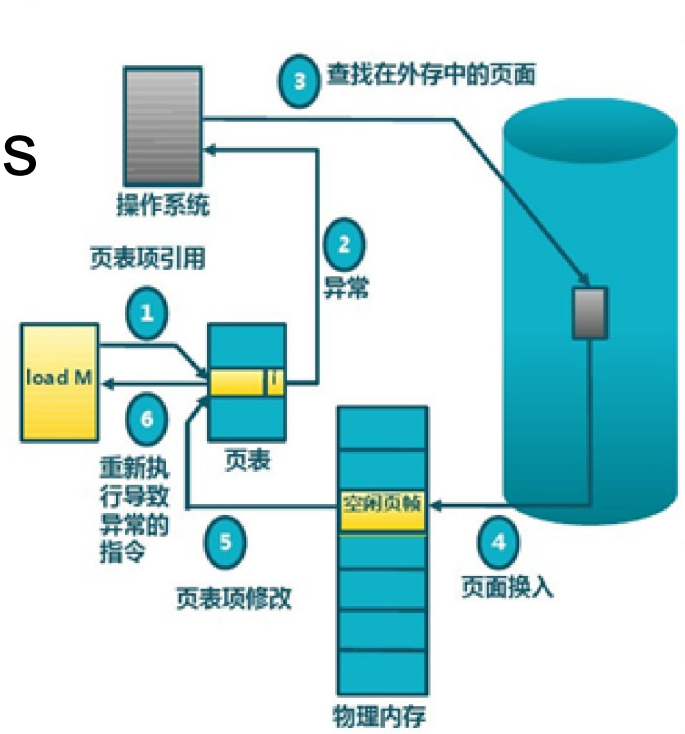
TLB	Page table	Cache	Possible? If so, under what circumstance?
hit	hit	miss	Possible, although the page table is never really checked if TLB hits.
miss	hit	hit	TLB misses, but entry found in page table; after retry, data is found in cache.
miss	hit	miss	TLB misses, but entry found in page table; after retry, data misses in cache.
miss	miss	miss	TLB misses and is followed by a page fault; after retry, data must miss in cache.
hit	miss	miss	Impossible: cannot have a translation in TLB if page is not present in memory.
hit	miss	hit	Impossible: cannot have a translation in TLB if page is not present in memory.
miss	miss	hit	Impossible: data cannot be allowed in cache if the page is not in memory.

x86例



虚存管理的性能：有效访存时间

- EAT：effective memory access time
  - 访存时间*（1-P）+ 缺页处理时间 * P
    - 缺页处理时间=读盘时间+写回时间*q
  - 访存时间10ns
  - 磁盘访问时间5ms
  - 缺页率P
  - Dirty率q



A.在内存中有空闲物理页面时，分配一物理页帧f，转第E步；

B.依据页面置换算法选择将被替换的物理页帧f，对应逻辑页q

C.如q被修改过，则把它写回外存；

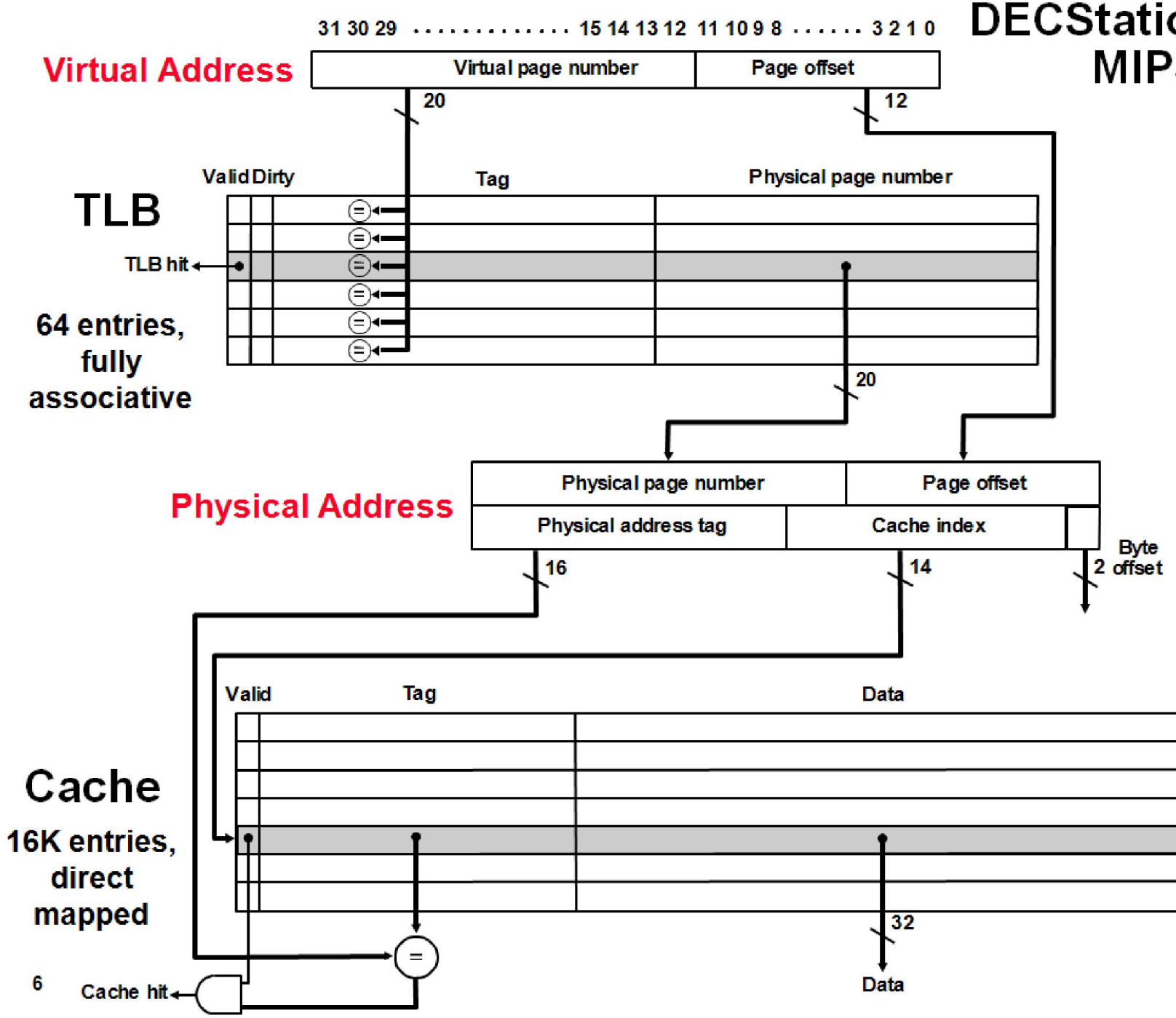
D.修改q的页表项中驻留位置为0；

E.将需要访问的页p装入到物理页面f

F.修改p的页表项驻留位为1，物理页帧号为f；

G.重新执行产生缺页的指令

DECStation 3100/ MIPS R2000



可用内存大小

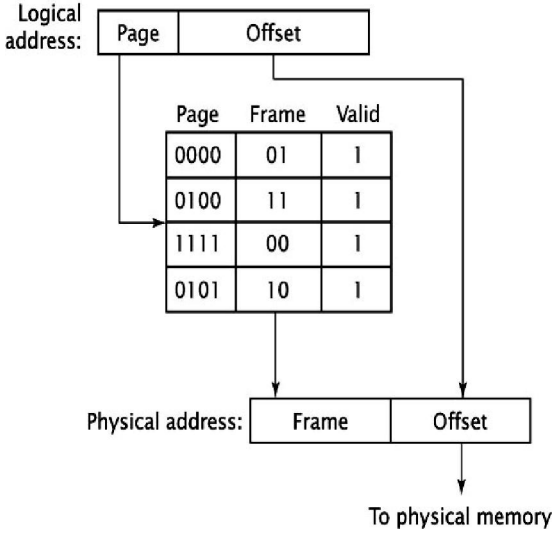
用free -m查看的结果：

```
[root@localhost ~]# free -m
             total       used       free     shared    buffers     cached
Mem:          7918       7865         52          0       7228        143
-/+ buffers/cache:        493       7424
Swap:         4996          0       4996
```

- 用 used 减去 buffer 和 cache，才是运行中的程序所占用的空间

小结

- 实方式下，程序空间与物理空间对应
- 虚方式下，每个进程有自己的虚空间
  - 虚存的主要作用？
  - 每个进程一个（套）页表，由OS创建进程时创建
  - 在辅存中保存进程的虚空间，主存为虚空间的部分镜像
- 作业：5.10.1，5.10.4
  - 选一
    - 虚存管理哪些由硬件实现，哪些由OS实现？
      - MMU中包含哪些模块？
    - TLB miss和缺页异常如何处理？
    - 执行一条load/store指令需要访存几次？
- 思考
  - 程序的虚地址空间与磁盘空间如何映射？
    - 即：swap区是如何组织的？
  - MMU控制器如何实现？
  - Cache与MMU实现机制比较？
  - 程序执行前，OS要做什么？
  - 系统启动时是实地址方式，何时转为虚方式？切换时发生了什么？




Thank You