Memory Management

Disclaimer: some slides are adopted from book authors' slides with permission

Roadmap

- CPU management
  - Process, thread, synchronization, scheduling
- **Memory management**
  - **Virtual memory**
- Disk management
- Other topics

Memory Management

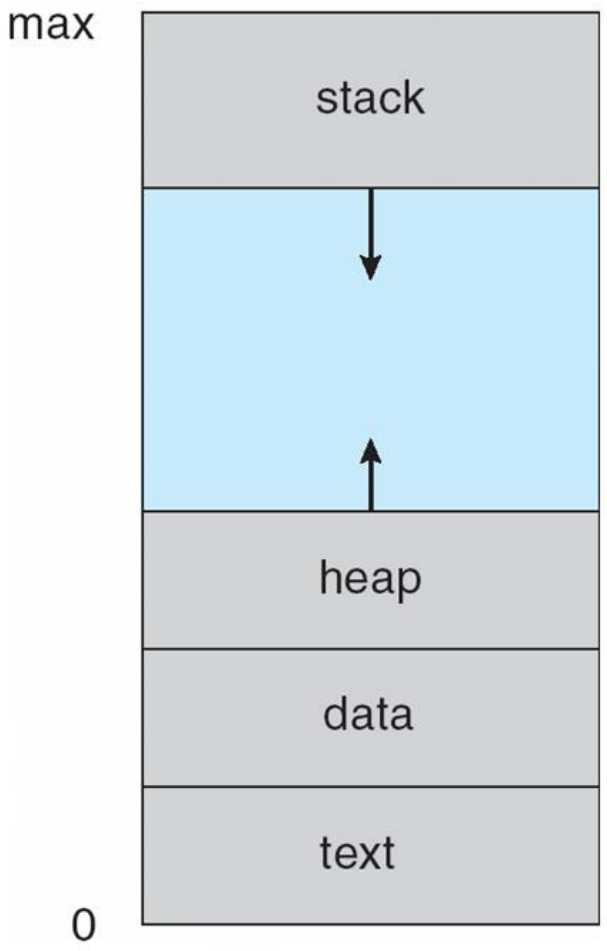
- Goals
  - Easy to use abstraction
    - Same virtual memory space for all processes
  - Isolation among processes
    - Don’t corrupt each other
  - Efficient use of capacity limited physical memory
    - Don’t waste memory

Concepts to Learn

- Virtual address translation
- Paging and TLB
- Page table management
- Swap

Virtual Memory (VM)

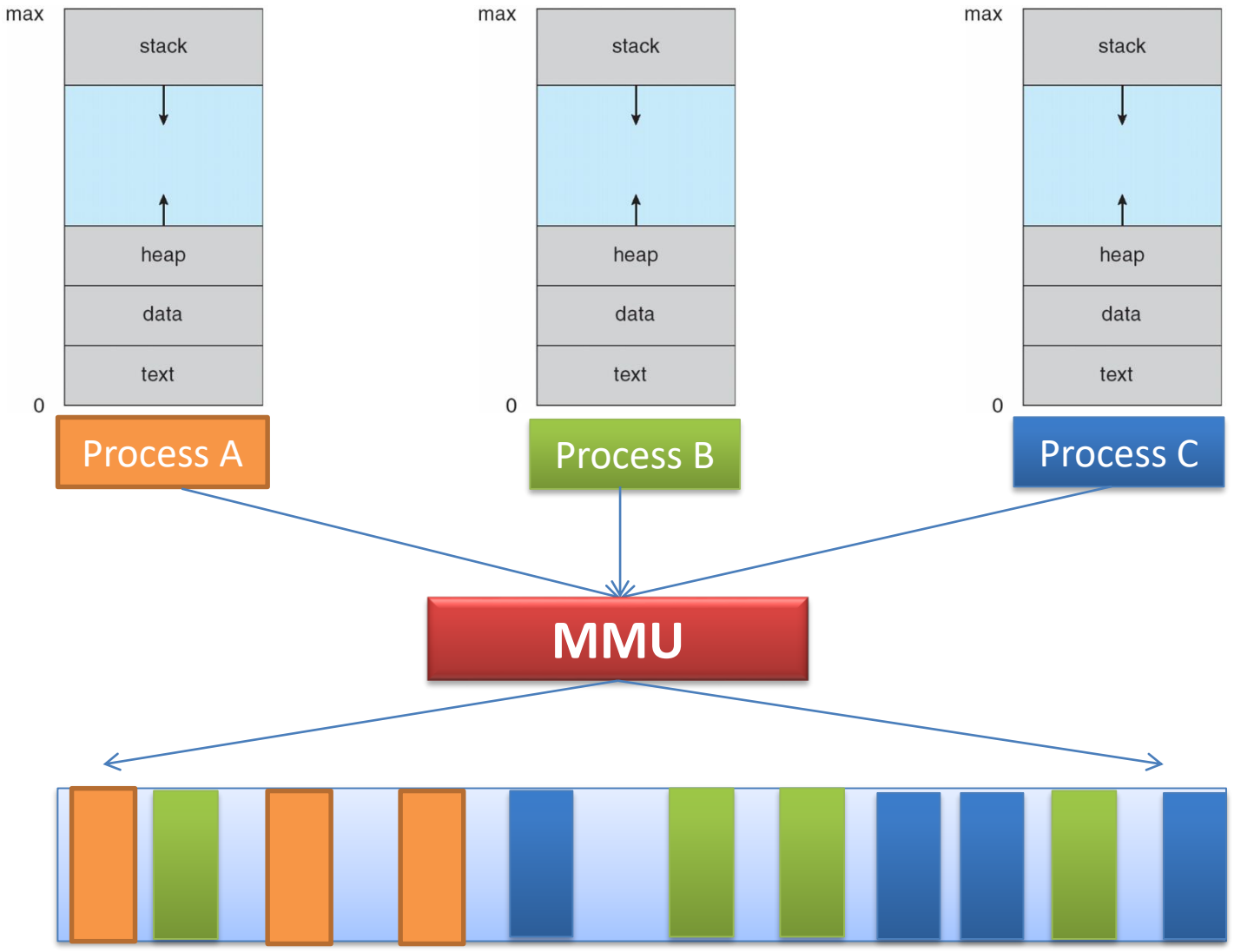
- Abstraction
  - 4GB linear address space for each process
- Reality
  - 1GB of actual physical memory shared with 20 other processes
- How?



Virtual Memory

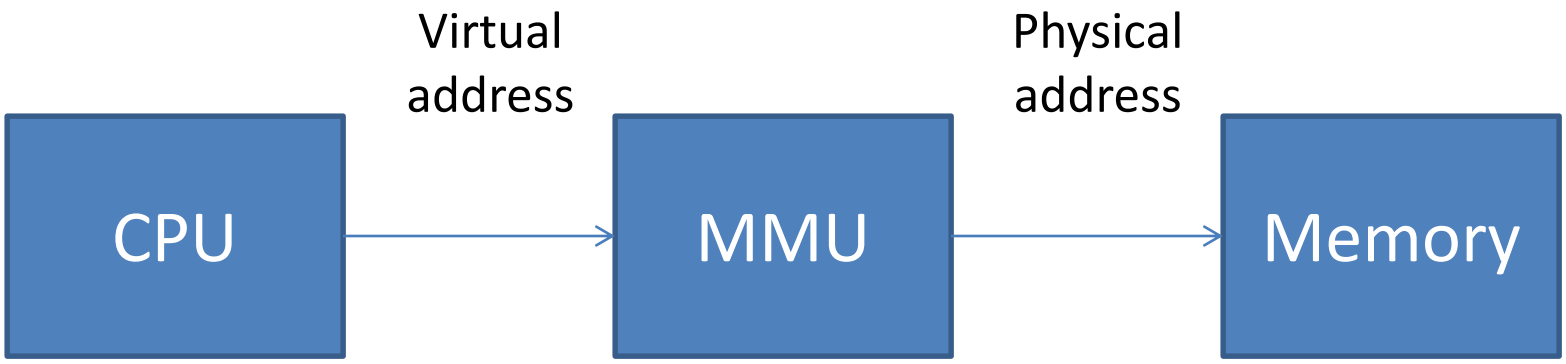
- Hardware support
  - MMU (memory management unit)
  - TLB (translation lookaside buffer)
- OS support
  - Manage MMU (sometimes TLB)
  - Determine address mapping
- Alternatives
  - No VM: many real-time OS (RTOS) don't have VM

Virtual Address



MMU

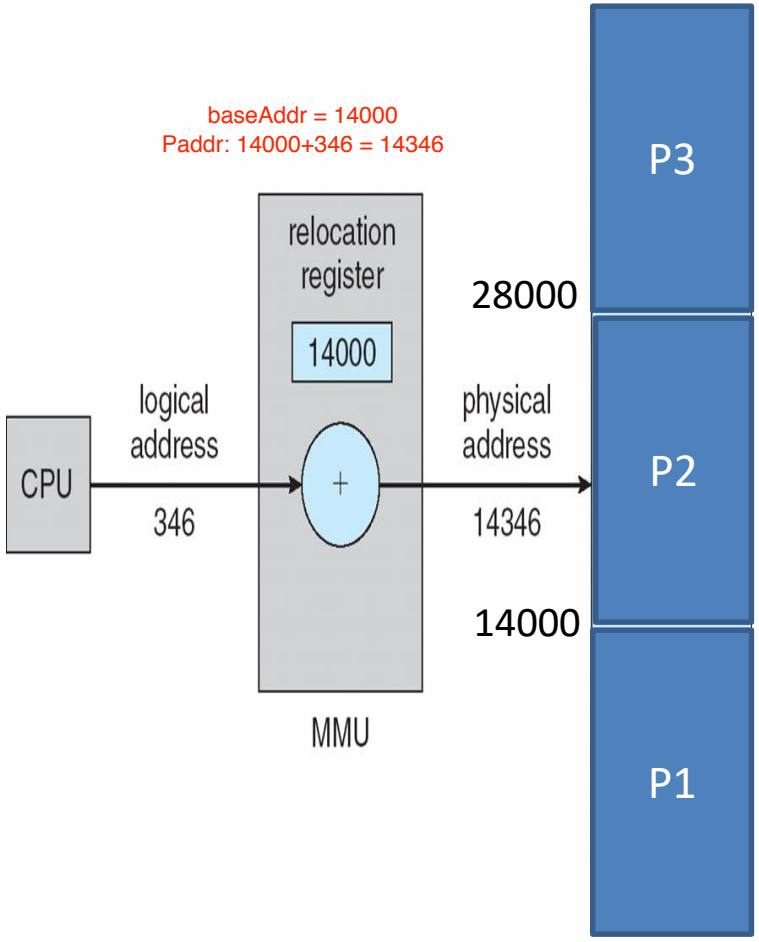
- Hardware unit that translates *virtual address* to *physical address*



A Simple MMU

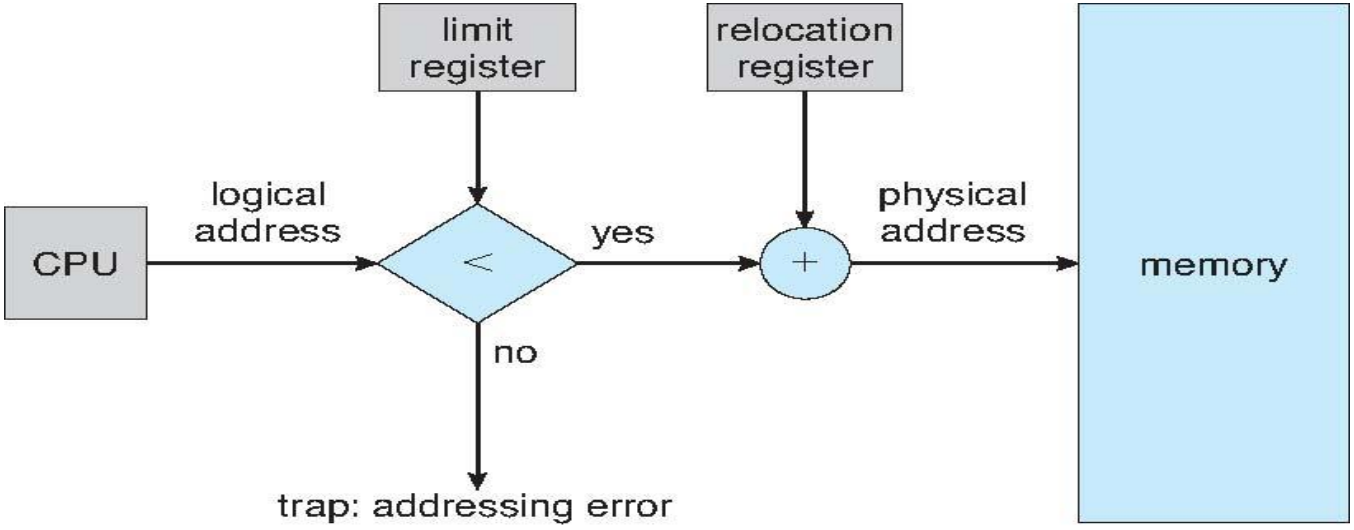
- BaseAddr: base register
- Paddr = Vaddr + BaseAddr

- Advantages
  - Fast
- Disadvantages
  - No protection
  - Wasteful



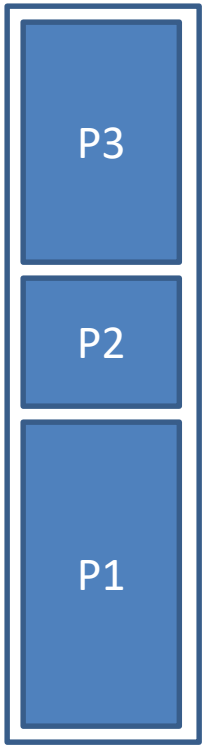
A Better MMU

- Base + Limit approach
  - If Vaddr > limit, then trap to report error
  - Else Paddr = Vaddr + BaseAddr



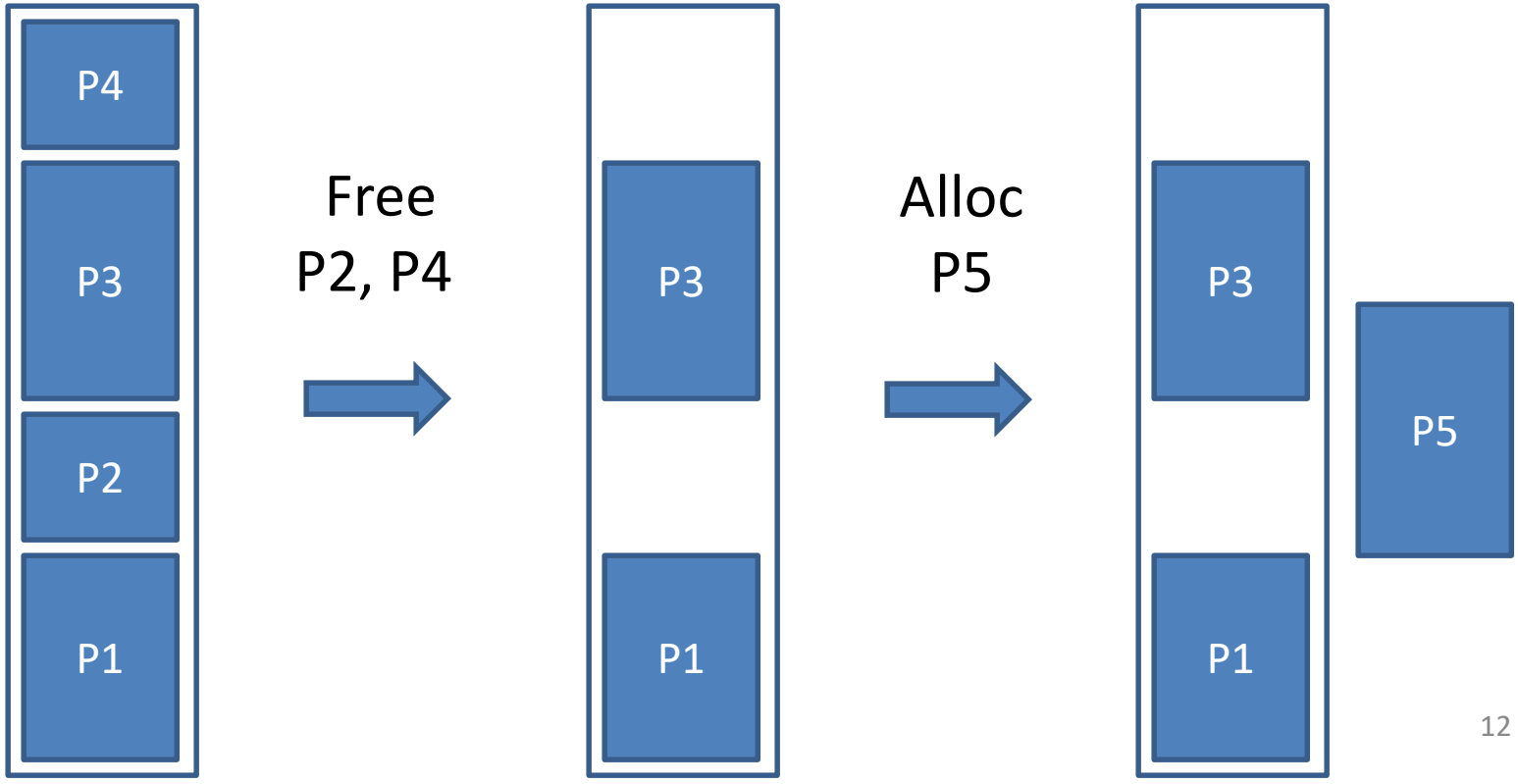
A Better MMU

- Base + Limit approach
  - If Vaddr > limit, then trap to report error
  - Else Paddr = Vaddr + BaseAddr
- Advantages
  - Support protection
  - Support variable size partitions
- Disadvantages
  - Fragmentation



Fragmentation

- External fragmentation
  - total available memory space exists to satisfy a request, but it is not contiguous 连续的



Modern MMU

- Paging approach

分页内存管理；非连续性的
页表结构：层次结构（multi），哈希结构（hash）和反向结构

  - Divide physical memory into fixed-sized blocks called frames (e.g., 4KB each)

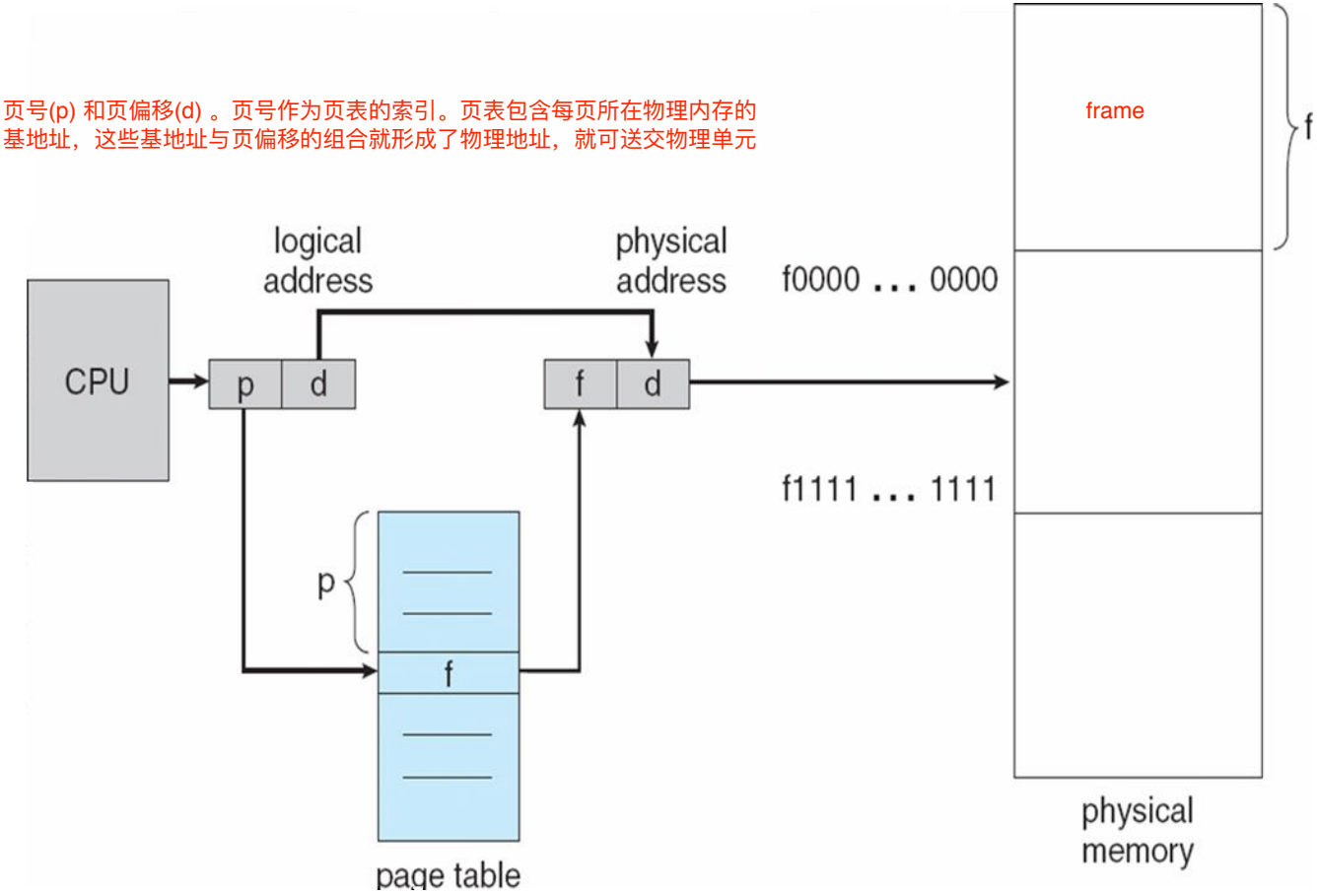
2^2 * 2^10 = 4KB
1*10^3 =1K

  - Divide logical memory into blocks of the same size called pages (page size = frame size)
  - Pages are mapped onto frames via a table ➔ page table

https://blog.csdn.net/qqqqq1993qqqqq/article/details/74199784

Modern MMU

- Paging hardware



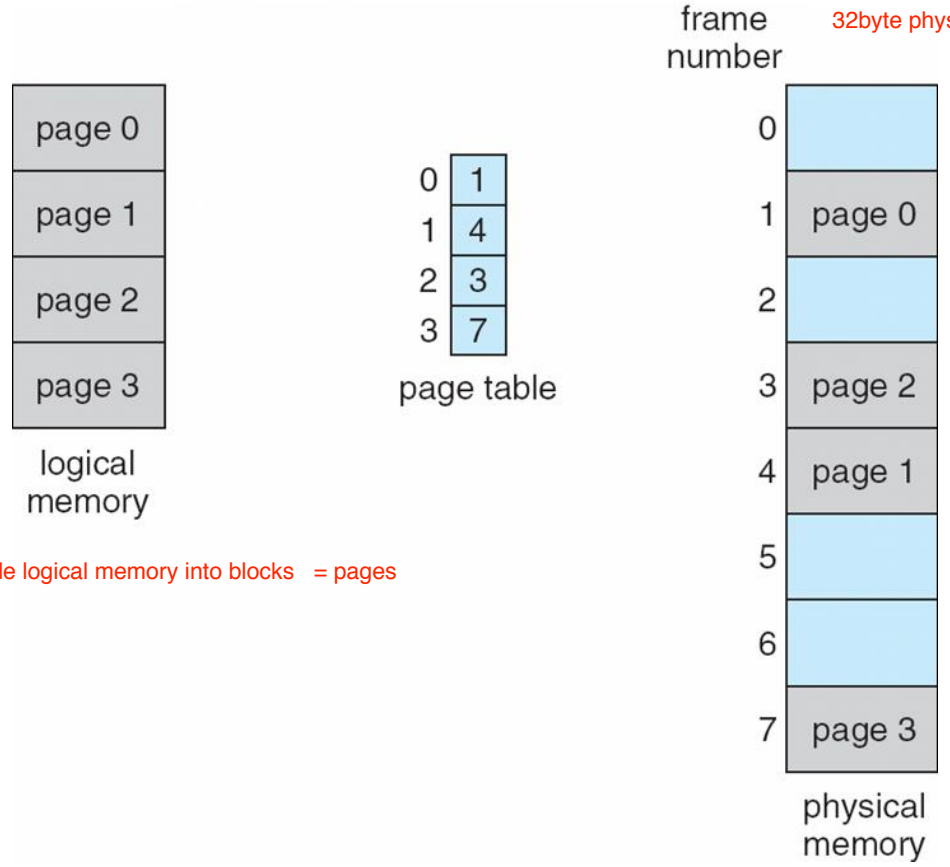
Modern MMU

n=2 m =4 page size = 4 bytes Memory = 32 bytes; so memory was divided into 8 part; Actually we have 4 pages

- Memory view

each page size == each frame size

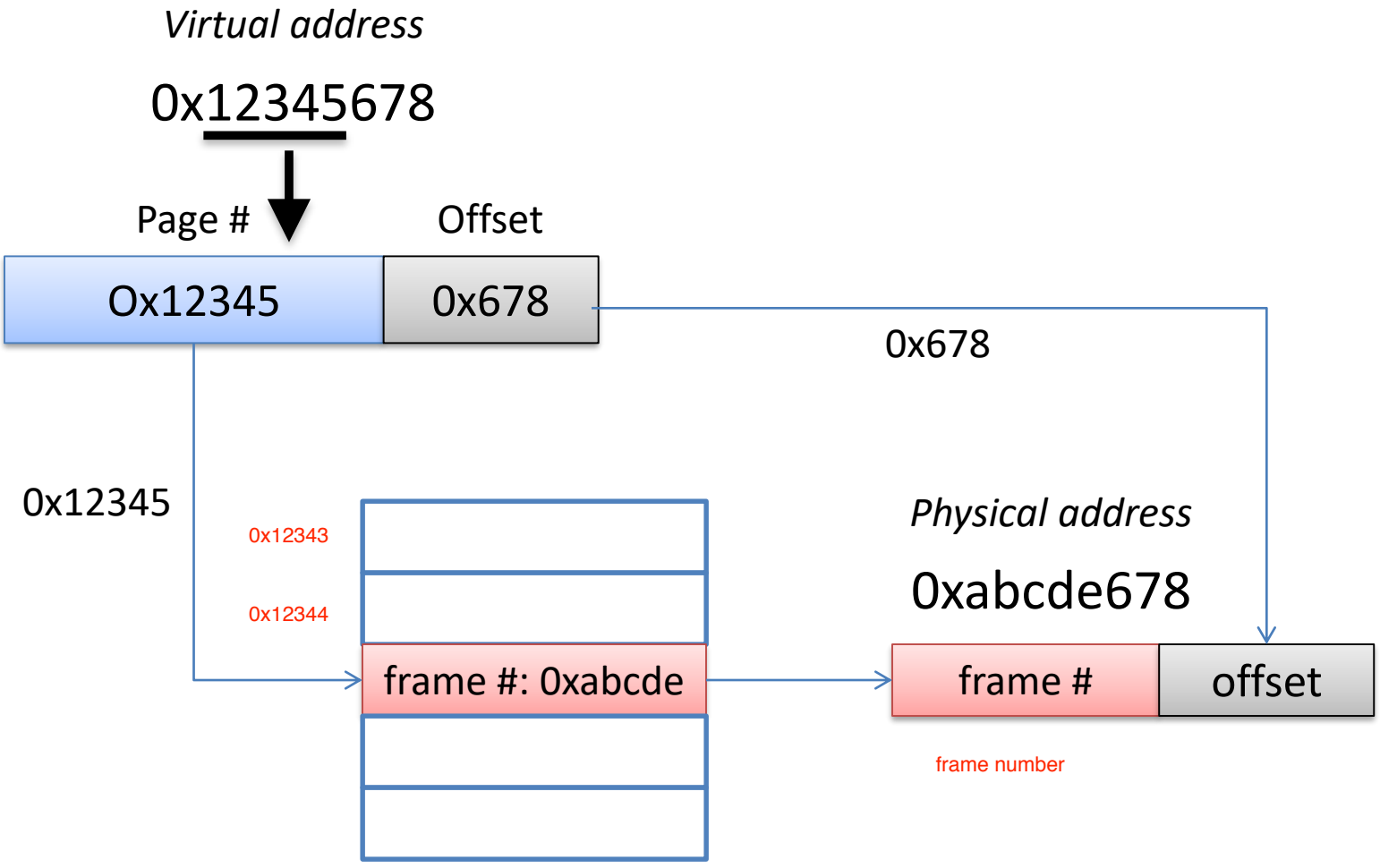
logical memory = n+ m
n 决定 page 数量
m决定页偏移量



divide logical memory into blocks = pages

32byte physical memory / 4 byte PTE = 8 pages

Virtual Address Translation



Advantages of Paging

- No external fragmentation
  - Efficient use of memory
  - Internal fragmentation (waste within a page) still exists

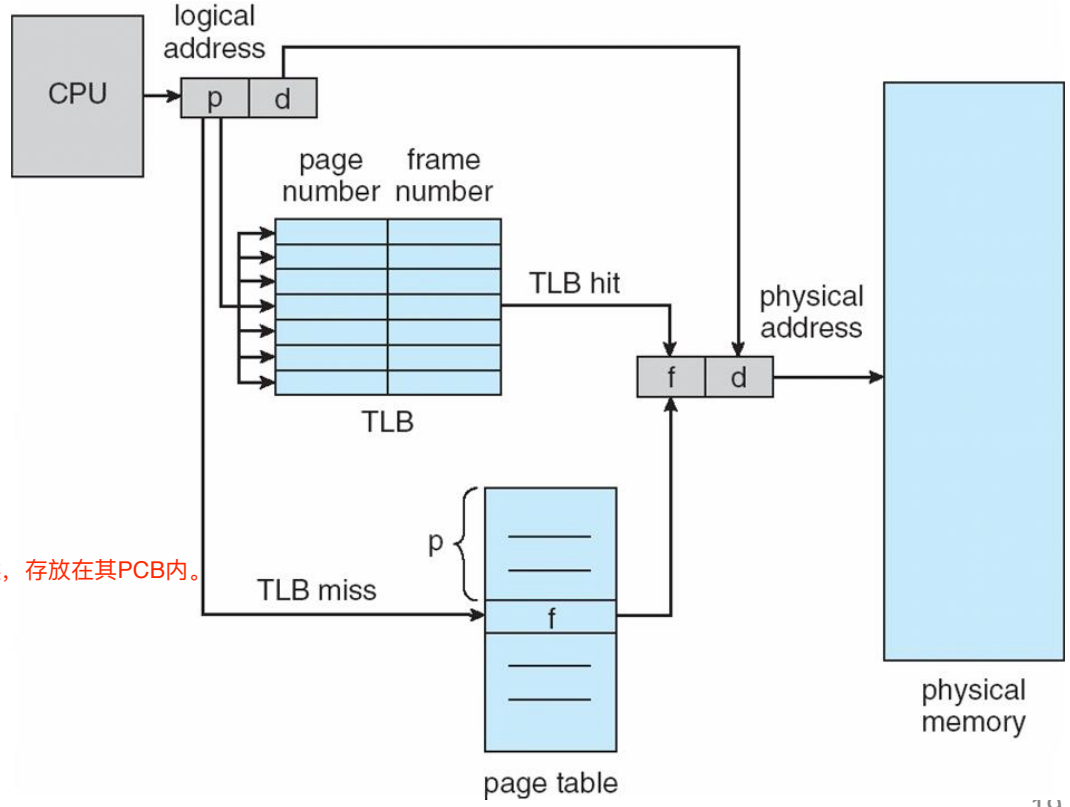
Issues of Paging

- Translation speed
  - Each load/store instruction requires a translation
  - Table is stored in **memory**
  - Memory is **slow** to access
    - ~100 CPU cycles to access DRAM

Translation Lookaside Buffer (TLB)

Important 后面会学到很多替换方法当TLB满了
现查找TLB的，再找Page table

- **Cache** frequent address translations
  - So that CPU don't need to access the page table all the time
  - Much faster



https://blog.csdn.net/u014338577/article/details/82750771

所以页表被放在物理内存中，由操作系统维护。
程都有页表，页表起始地址和页表长度的信息在进程不被CPU执行的时候，存放在其PCB内。

Issues of Paging

Memory内存基本单元：Byte

Since we have a virtual address space of 2^32 and each page size is 2^12, we can store (2^32/2^12) = 2^20 pages. Since each entry into this page table has an address of size 4 bytes, then we have 2^20*4 = 4MB. So the page table takes up 4MB in memory.

- Page size
  - Small: minimize space waste, requires a large table
  - Big: can waste lots of space, the table size is small
  - Typical size: 4KB

32位操作系统最低12位的页偏移量—>所以PTE =2^12 * 1B = 4KB

  - How many pages are needed for 4GB (32bit)?
    - 4GB/4KB = 1M pages

B —>KB—>MB—>GB(GigaBytes) => 2^32/2^12 = 2^20
4GB/4KB == 1(M) =2^20

  - What is the required page table size?
    - assume 1 page table entry (PTE) is 4bytes
    - 1M * 4bytes = 4MB

4MB page table size can represent 4GB spaces

  - Btw, this is for each process. What if you have 100 processes? Or what if you have a 64bit address?

100 * 4MB = 400 MB too large
2^64/2^12 = 2^52 too large

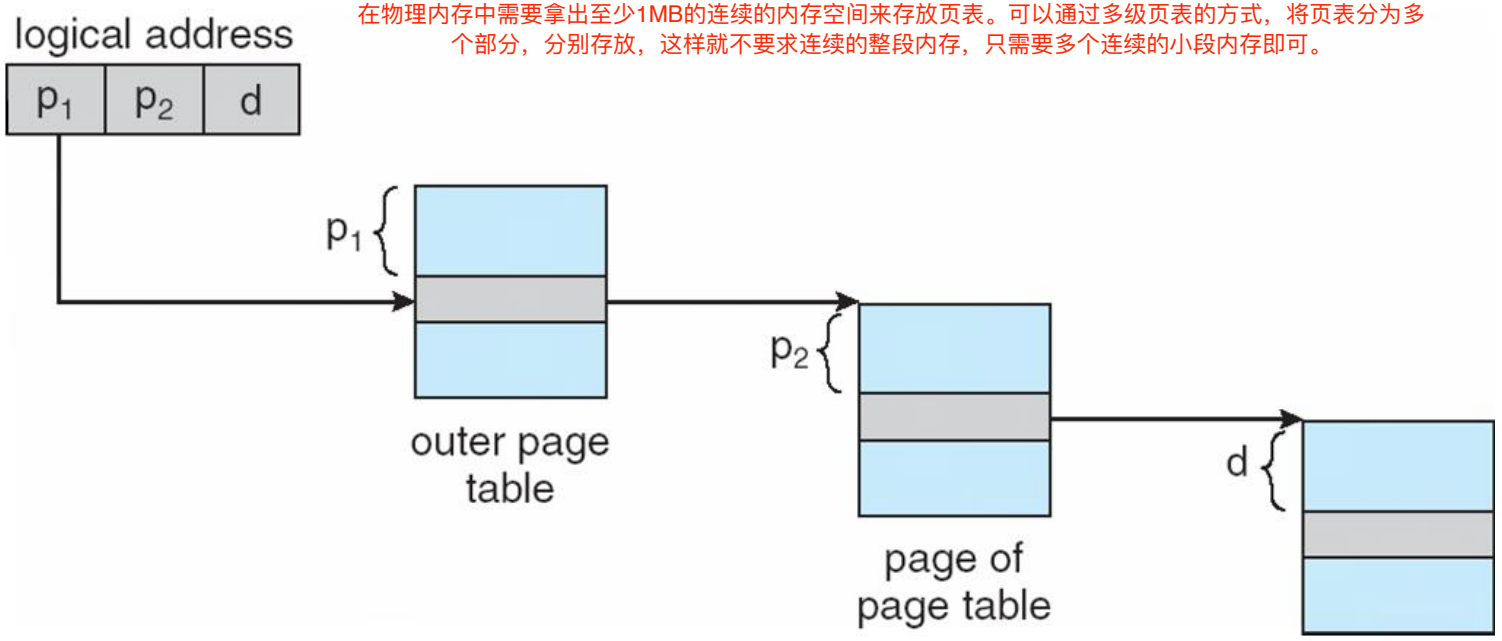
Paging

- Advantages
  - No external fragmentation
- Two main Issues
  - Translation speed can be slow
    - TLB
  - Table size is big

Multi-level Paging

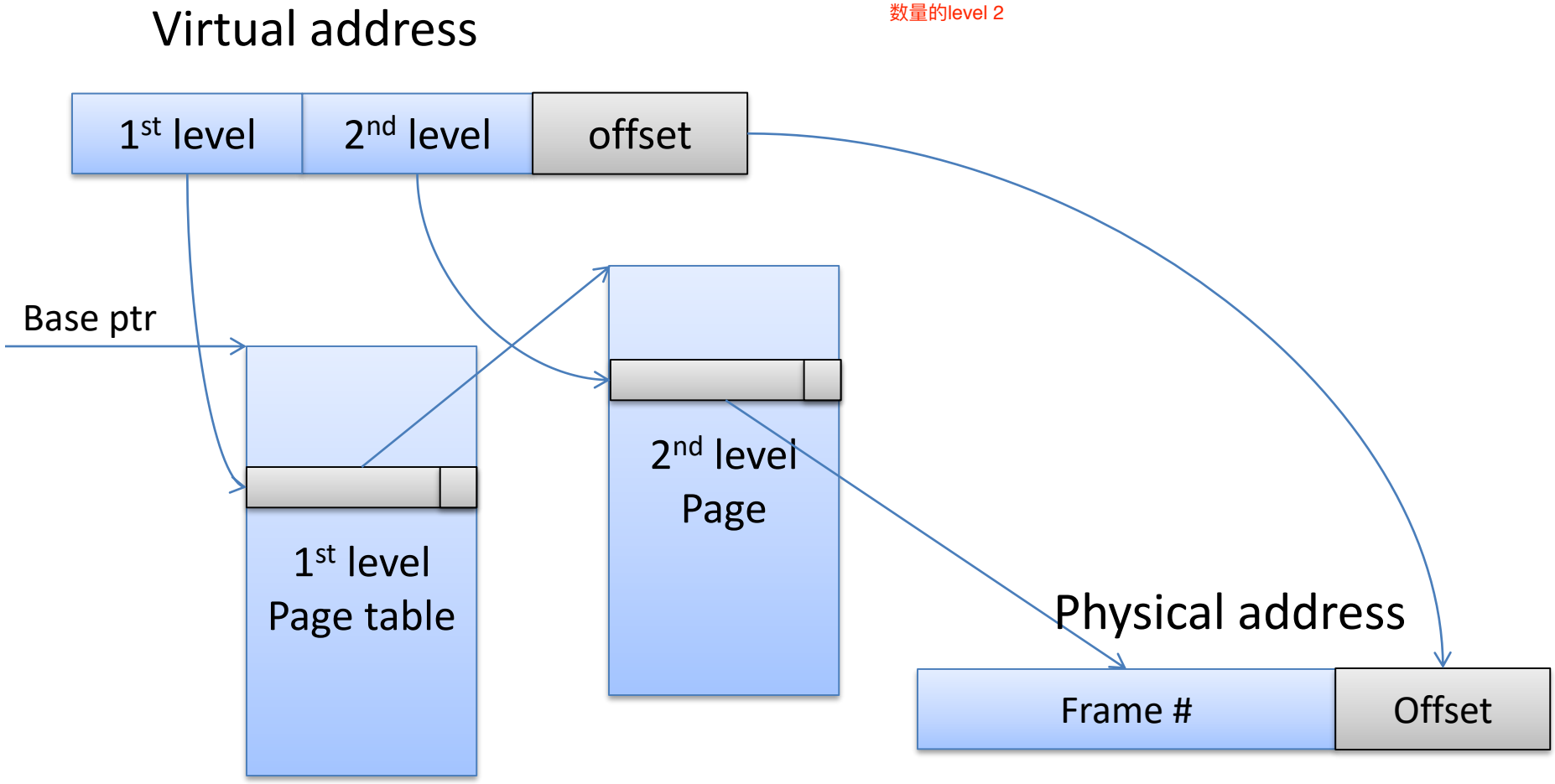
- Two-level paging

在物理内存中需要拿出至少1MB的连续的内存空间来存放页表。可以通过多级页表的方式，将页表分为多个部分，分别存放，这样就不要求连续的整段内存，只需要多个连续的小段内存即可。



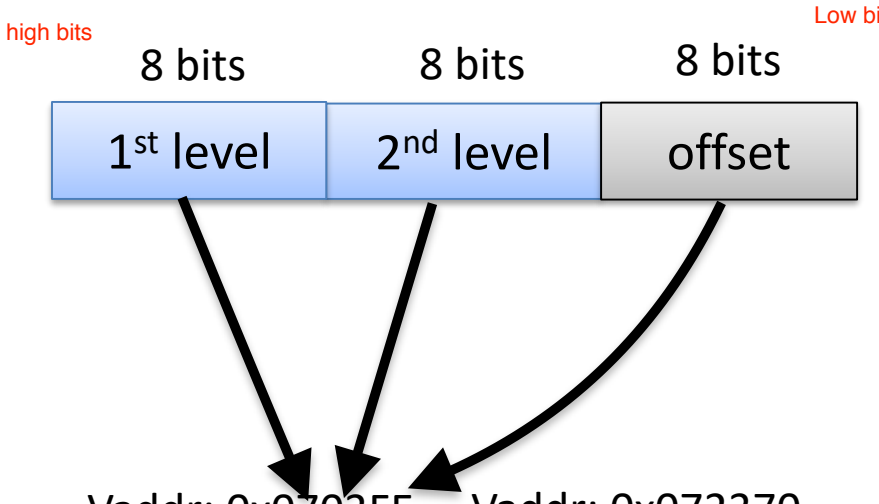
Two Level Address Translation

第一层所有指针不是全部被使用，根据内存的不同，使用不同数量的level 2



1st level 里面存的是level 2 中第几个page

Example



Virtual address format (24bits)

Vaddr: 0x0703FE
1st level idx: 07
2nd level idx: 03
Offset: FE

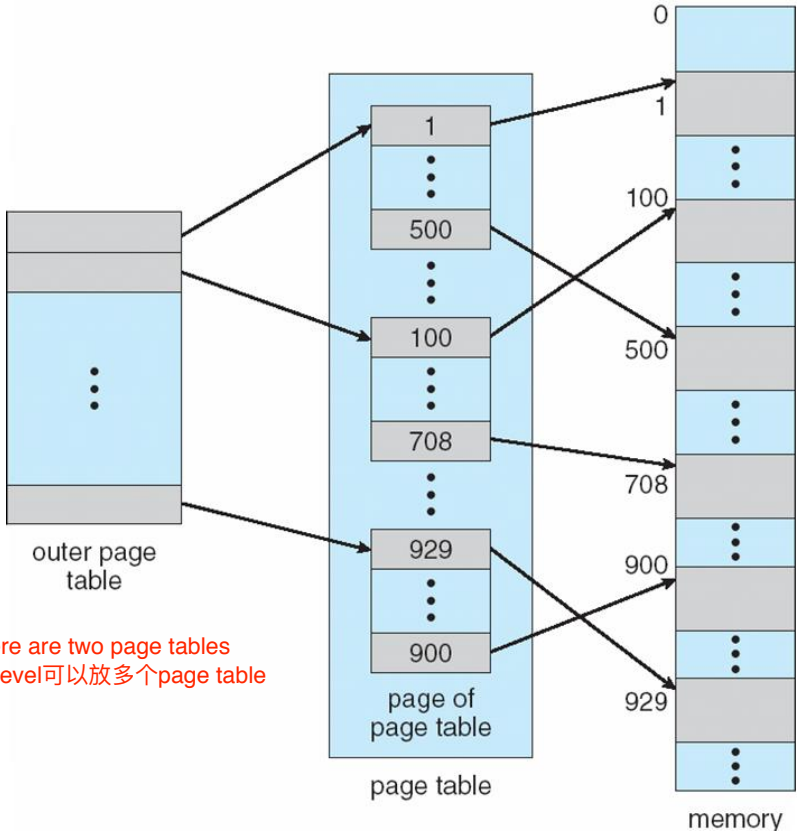
Vaddr: 0x072370
1st level idx: 07
2nd level idx: 23
Offset: 70

Vaddr: 0x082370
1st level idx: 08
2nd level idx: 23
Offset: 70

Multi-level Paging

- Can save table space
- How, why?

- Don’t need to create all mappings in the outer page table



There are two page tables
第二个level可以放多个page table

Summary

- MMU
  - Virtual address → physical address
  - Various designs are possible, but
- Paged MMU
  - Memory is divided into fixed-sized pages
  - Use page table to store the translation table
  - No external fragmentation

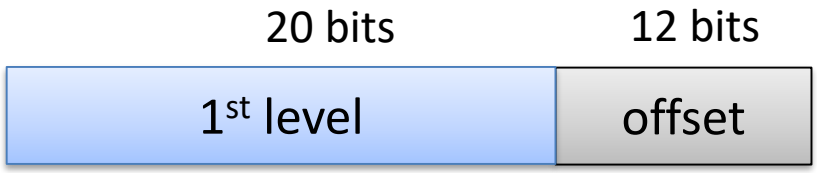
Summary

- Paged MMU: Two main Issues
  - Translation speed can be slow
    - TLB
  - Table size is big
    - Multi-level page table

Quiz

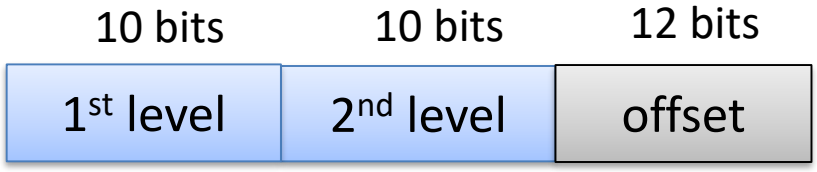
- What is the minimum page table size of a process that uses only 4MB memory space?
  - assume a PTE size is 4B

number of pages * PTE



page number

4 * 2^20 = 4MB



4 * 2^10 + 4* 2^10 = 8KB

Quiz

- What is the page table size for a process that only uses 8MB memory?
  - Common: 32bit address space, 4KB page size
  - Case 1) 1-level page table
    - Assume each page table entry is 4 bytes
    - Answer: 2^20 x 4 byte = 4MB
      1M
  - Case 2) two-level page table
    - Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both-levels, single page table entry size is 4 bytes
    - Answer: 2^10 x 4 + 2 x (2^10 x 4) = 4KB + 8KB = 12KB

Quiz

- What is the page table size for a process that only uses 16MB memory?
  - Common: 32bit address space, 4KB page size
  - Case 1) 1-level page table
    - Assume each page table entry is 4 bytes
    - Answer: 2^20 x 4 byte = 4MB
  - Case 2) two-level page table
    - Assume first 10 bits are used as the index of the first-level page table, next 10 bits are used as the index of the second-level page table. In both-levels, single page table entry size is 4 bytes
    - Answer: 2^10 x 4 + 4 x (2^10 x 4) = 4KB + 16KB = 20KB