CS 4410
Operating Systems

Memory: Paging

Summer 2016

Cornell University

Today

- An allocation and protection mechanism that is used on most operating systems.

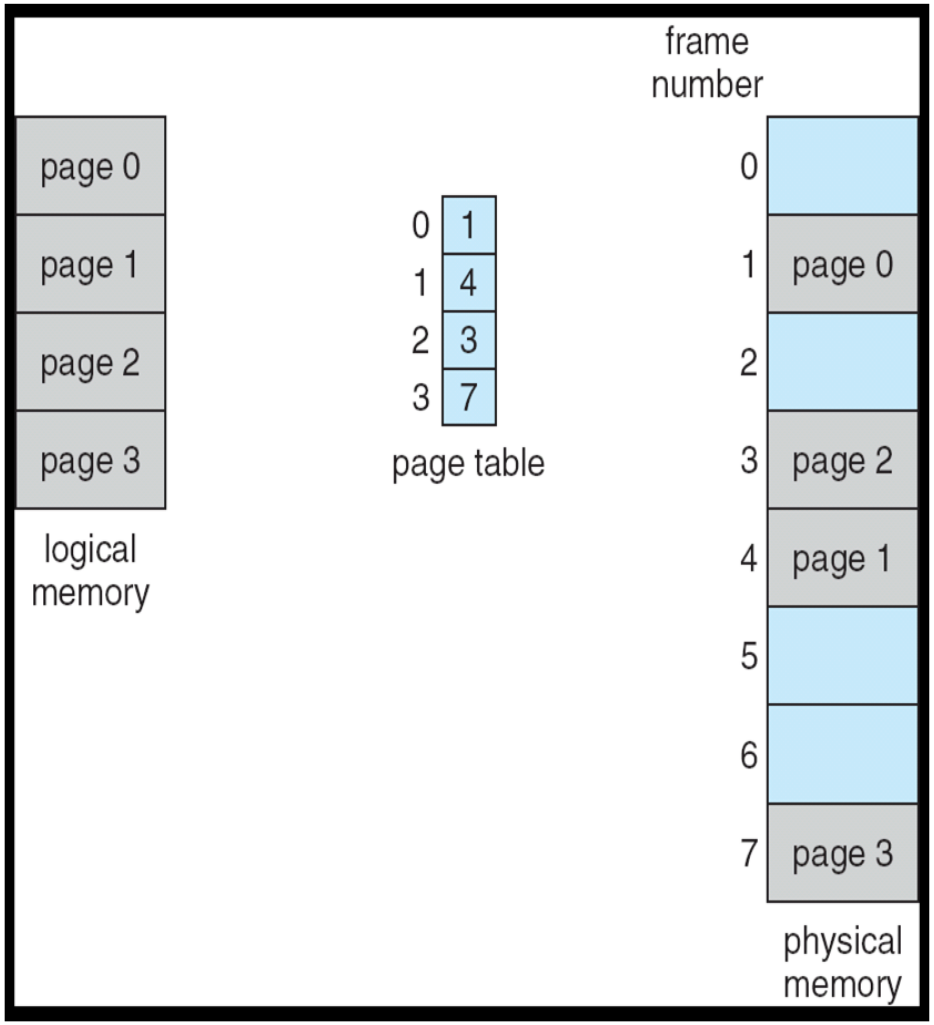
Paging

- Allocation and protection scheme.
- Processes get non-contiguous memory space.
- Memory is partitioned into fixed-size blocks.

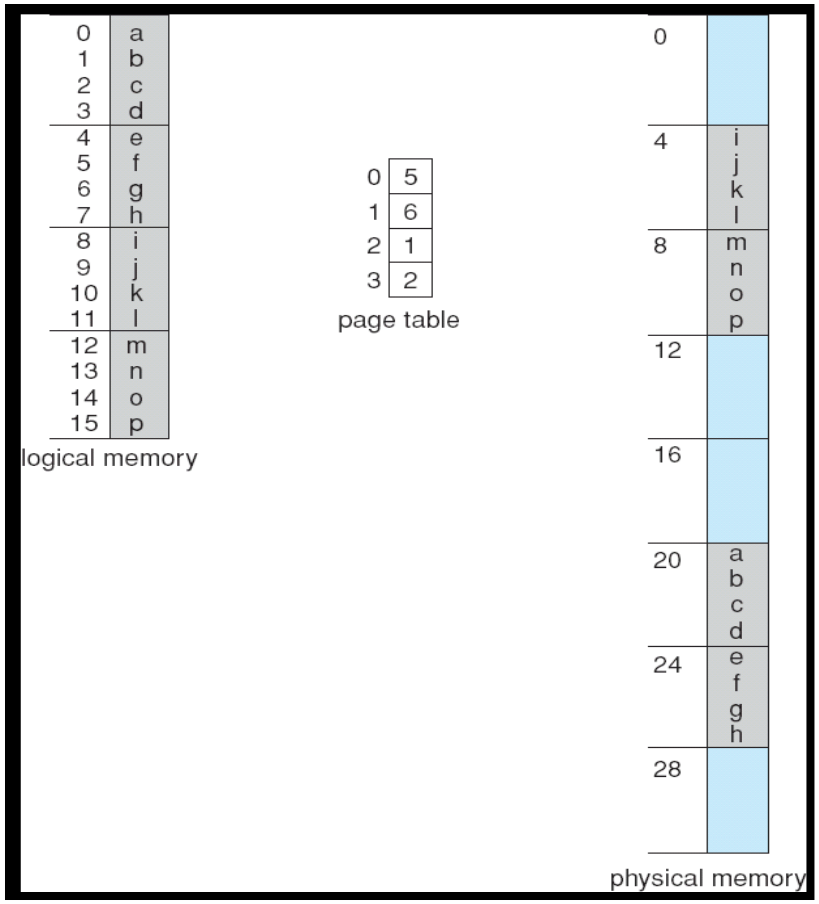
Paging

- Divide physical memory into **frames:**
  - Fixed-sized blocks.
  - Size is power of 2, between 512 bytes and 8,192 bytes.
- Divide virtual memory into **pages**.
  - Same size as frames.
- **Page table** translates virtual to physical addresses.

Paging


frame number
page 0
page 1
page 2
page 3
logical memory
0 1
1 4
2 3
3 7
page table
0
1 page 0
2
3 page 2
4 page 1
5
6
7 page 3
physical memory

Paging Example



Frame size = page size = 4 bytes

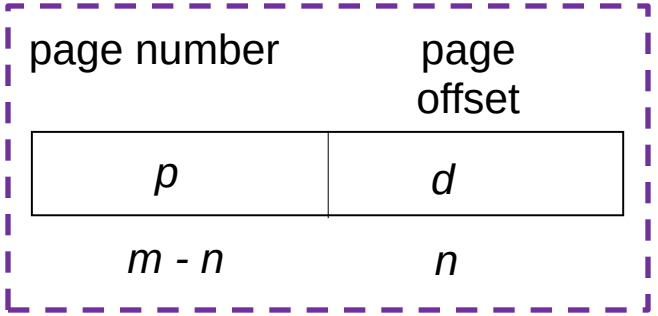
If CPU produces virtual address 11, what is the corresponding physical address?

- Find the page that 11 belongs to:
  - Page number:2
- Find the offset of 11 in page 2:
  - Page offset: 3
- Find the frame that 11 corresponds to:
  - Frame number:1
- Find the physical address that 11 corresponds to:
  - Physical address:

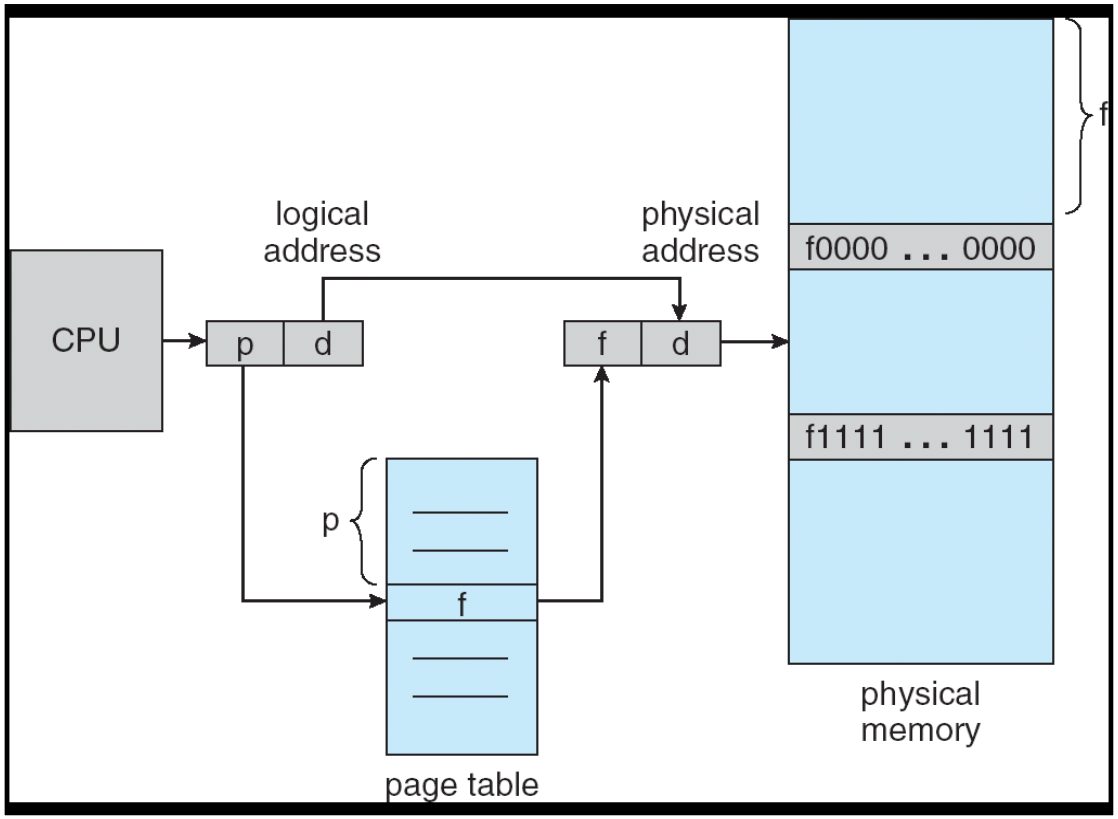
  Frame number * frame size + page offset = 7

Address Translation

- **Page number (p)** – used as an *index* into a page table which contains base address of each page in physical memory.
- **Page offset (d)** – combined with base address to define the physical memory address that is sent to the memory unit.
- We can find the page number and the page offset of a virtual address, if we know the size of pages.
- If virtual address v has m bits (virtual address space 2^m), and if the size of pages is 2^n, then:
  - the n least significant bits of v are used as d,
  - the remaining bits of v are used as p.



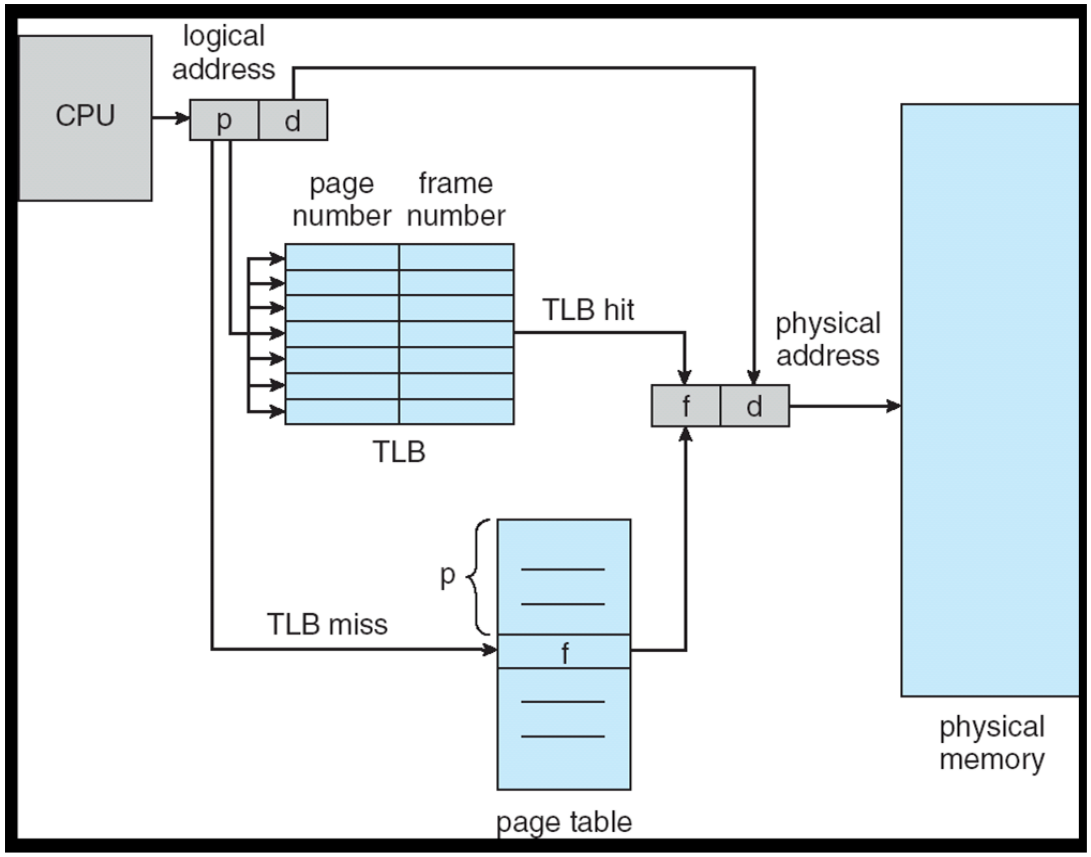
Address Translation Scheme



Accelerating translation with TLB

- Page tables are hold in memory.
  - So, every data/instruction access requires 2 memory accesses.
  - Memory accesses are much slower than instruction execution in CPU.
- To accelerate the translation mechanism, a special, small, fast-lookup hardware cache is added close to CPU.
- This translation look-aside buffer (TLB) contains a few (the most common) of the page-table entries.
- If page_number is in TLB get frame_number out.
  - No need to access the memory.

Paging Hardware With TLB

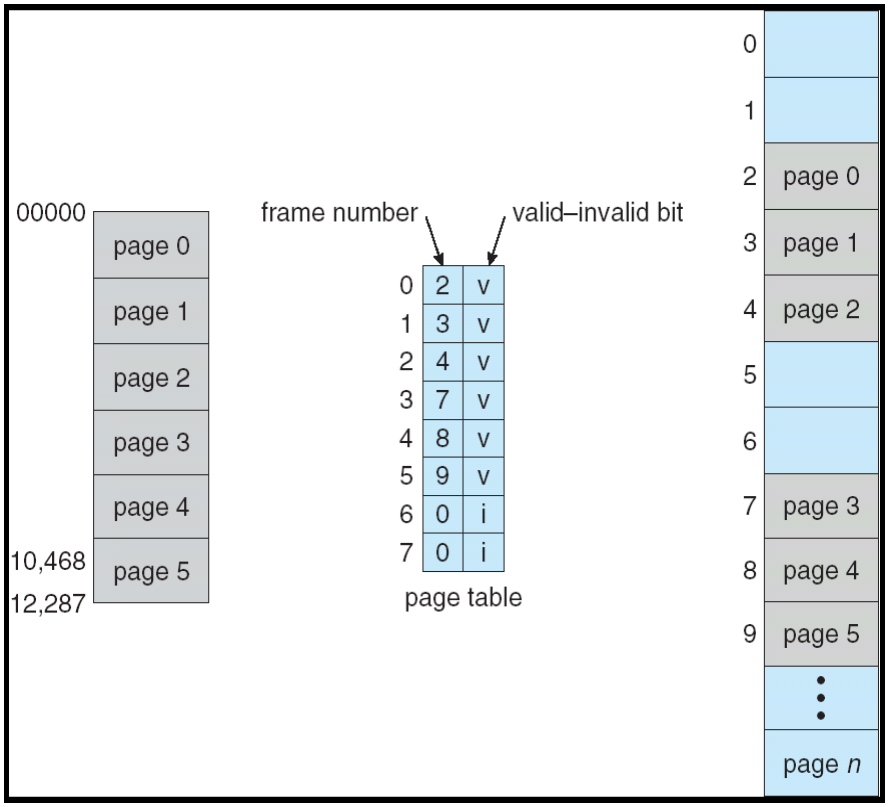


Updated Context Switch

- Save current process' registers in PCB.
- Set up Page Table Base Register (PTBR).
  - This is kept in PCB.
- Flash TLB.
- Restore registers of next process to run.
- Return from interrupt.

Memory Protection

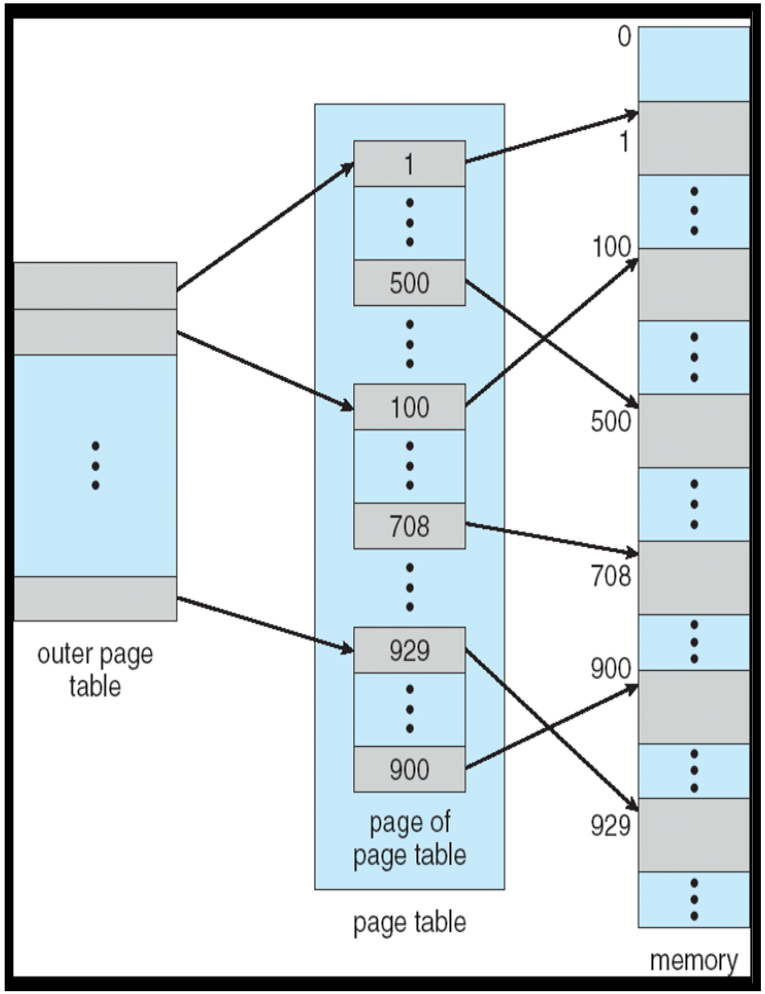
Implemented by associating protection bit with each frame.



Paging the Page Table

- A page table can get big.
  - It may need more than one page to be saved.
- So, linear searching of an address in the page table is not efficient.
- Need structures for efficient lookup of an address in the page table.

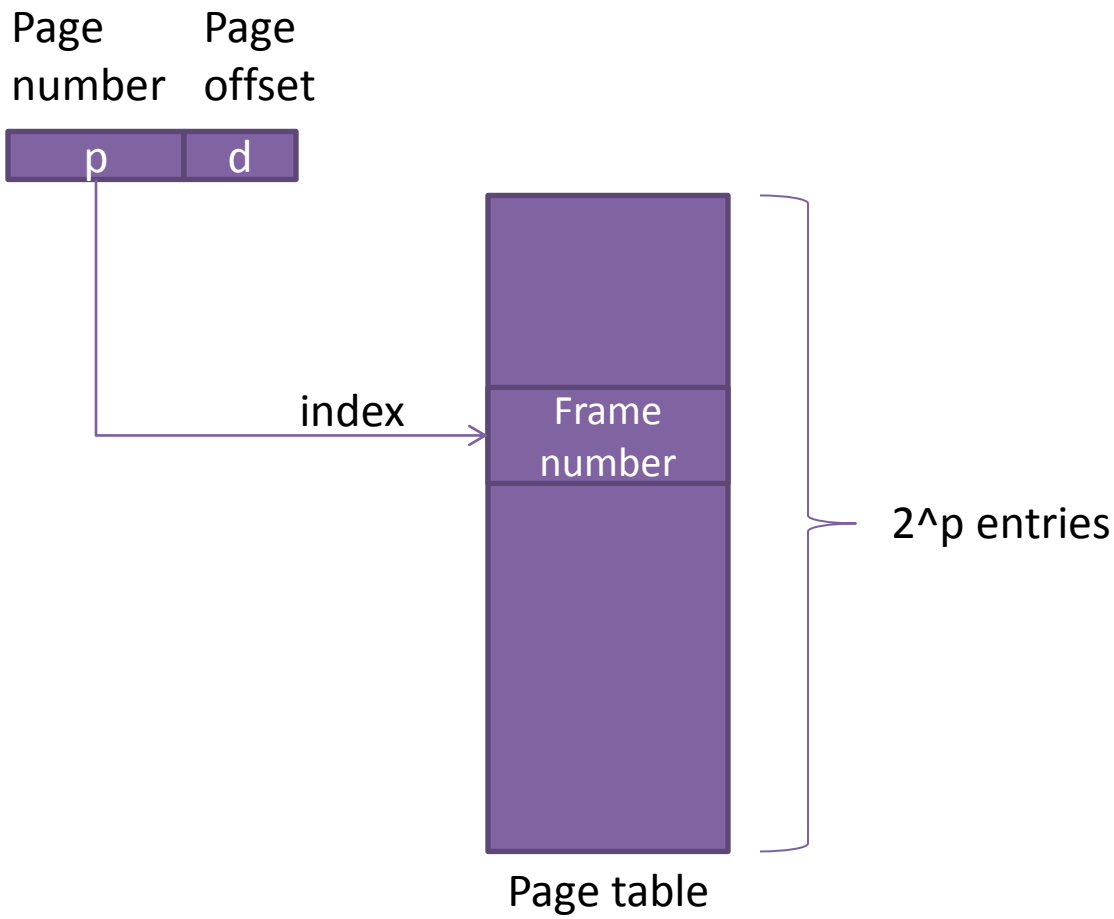
Hierarchical Paging



Hierarchical Paging

- 2^m bytes physical memory space and 2^n bytes virtual memory space.
  - Size of physical address: m bits
  - Size of virtual address: n bits
- Size of page: 2^d bytes.
  - Page offset of a virtual or physical address: d bits (the least significant)
  - Page number of a virtual address : n-d bits (the most significant) … say p bits
  - Frame number of a physical address: m-d bits
- A page table entry should fit a frame number and a valid bit, and it should be a power of 2.
  - Size e of page table entry ≥ m-d+1 bits (for simplicity e may be just m)
  - Entries per page: 2^d/(e/8) = 2^d/(m/8) … say 2^q entries/page
- Number of page table entries: 2^p
  - Number N' of pages for the page table: (2^p)/(2^q) = 2^(p-q)
- If N'>1, then we need hierarchical paging, otherwise, the page table fits in one page.

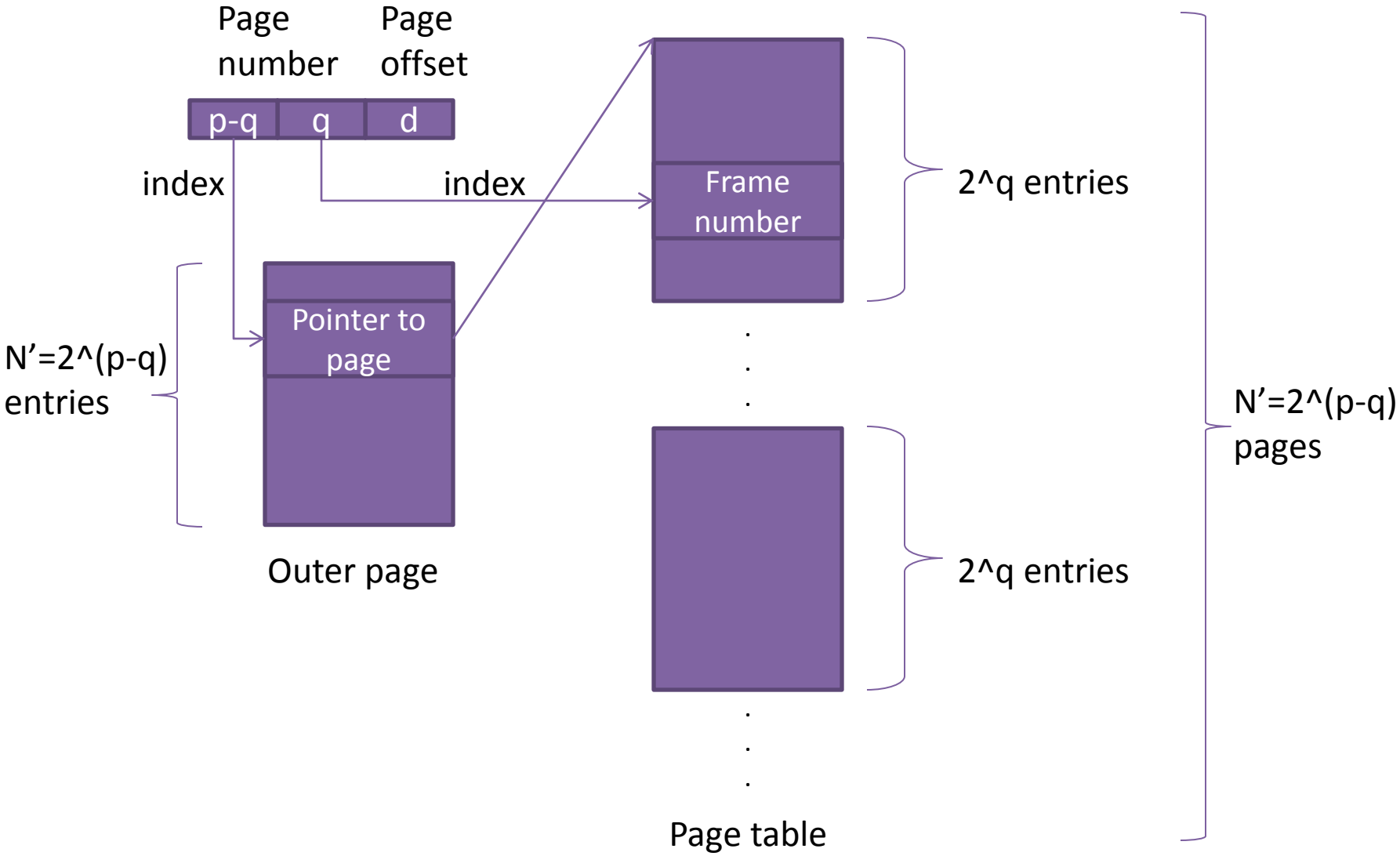
Paging: N'=1


Page number
Page offset
p
d
index
Frame number
2^p entries
Page table

Hierarchical Paging: N'>1

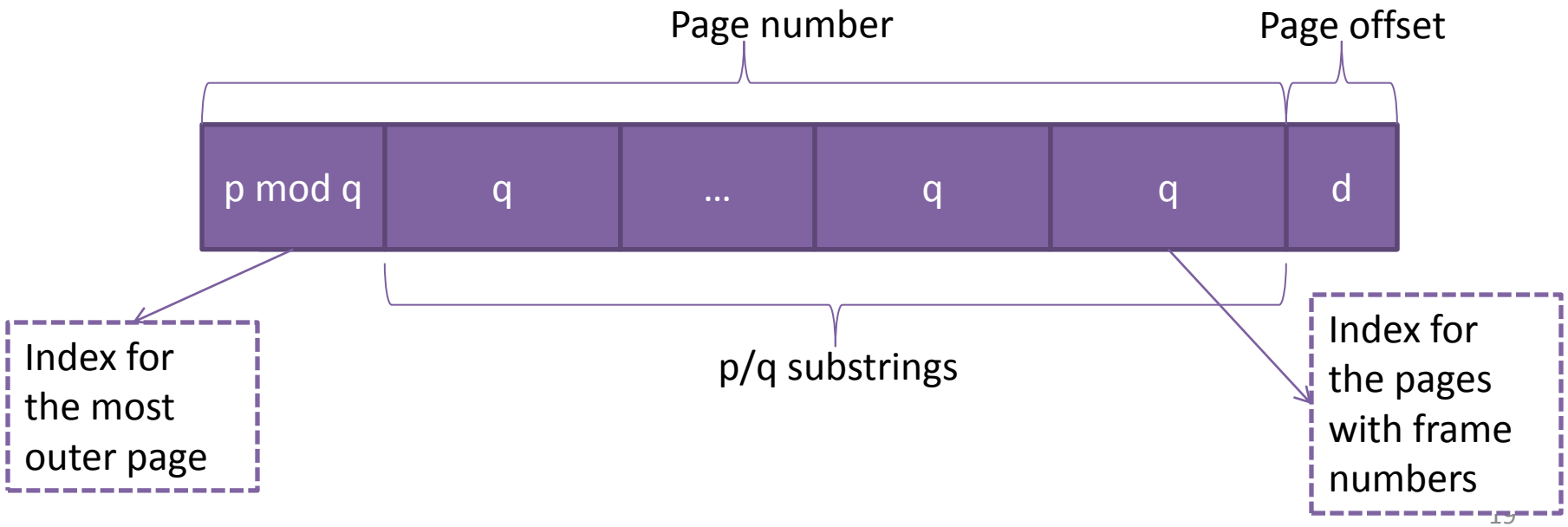
- Need outer pages that point to the N' pages.
- An outer page entry should fit a pointer to a page, so it should fit a physical address.
  - Size e' of outer page entry = m bits
  - Entries per page: 2^d/(m/8) … say 2^q entries/page
- Number of outer page entries: N'
  - Number N'' of pages for the outer page table: N'/(2^q) = 2^(p-q)/2^q = 2^(p-q-q)
- If N''>1, then we need to add one more level of outer pages…
  - Repeat the above steps.

Hierarchical Paging: N'>1, N''=1



Hierarchical Paging

- Just looking at a virtual address v and knowing the size of physical addresses (m bits), and the size of a page (2^d bytes), we can deduce:
  - how may levels the hierarchical paging is,
  - what is the page offset of v (d bits), what is the page number of v (p bits),
  - what substring pi of the page number is used as an index in the ith level of the hierarchy.
- Assume simple paging (N'=1) is a hierarchy with one level.
- Let 2^q = 2^d/(m/8). (entries per page)
- In this example, we need a hierarchy with: (/ is integer division)
  - (p/q)+1 levels, if p mod q ≠ 0, or
  - p/q levels, if p mod q = 0



Today

- Paging: an allocation and protection mechanism that is used on most operating systems.
- TLB
- Hierarchical paging

Coming up…

- Next lecture: Page replacement
- HW3
- Next in-class exam: Wednesday July 27th.