Operating Systems
Memory Management

Christian Khoury

Objectives

- Understand the various memory-managment mechanisms
  - Contiguous
  - Paging
  - Segmentation

Contents

Definitions

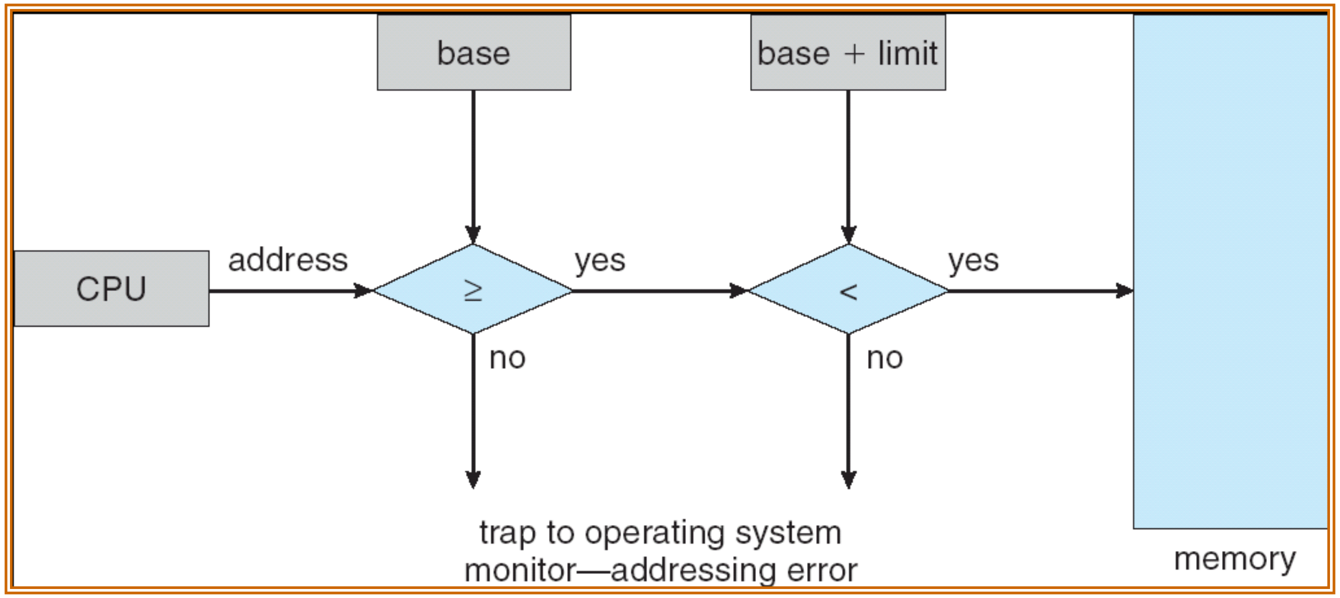
- Logical/Virtual Address
  - Generated by the CPU
- Physical Address
  - Generated by the MMU

Contents

Contiguous Allocation (1/5)

- Base register contains value of smallest physical address
- Limit register contains range of logical addresses – each logical address must be less than the limit register
- MMU maps logical address *dynamically*

Contiguous Allocation (2/5)



Contiguous Allocation (3/5)

- Multiple-partition allocation
  - Hole – block of available memory; holes of various size are scattered throughout memory
  - When a process arrives, it is allocated memory from a hole large enough to accommodate it
  - Operating system maintains information about: a) allocated partitions b) free partitions (hole)

Contiguous Allocation (4/5)

- Dynamic memory allocation strategies
  - First Fit
    - First on route“big enough” hole is used
  - Best Fit
    - Allocate the smallest hole that is big enough
    - Smallest leftover holes
  - Worst Fit
    - Allocate the largest hole that is big enough
    - Largest leftover holes

Contiguous Allocation (5/5)

- External Fragmentation
  - Processes are loaded an removed from memory
  - Free memory (holes) is broken into little pieces (Fragments)
  - May have enough global memory for a request but noncontiguous !
- Solutions
  - Compaction : Not always possible, expensive
  - Paging, Segmentation

Contents

Paging (1/11)

- Memory managment scheme that permits the physical address space to be noncontiguous/fragmented

- Need to map logical addresses into physical ones

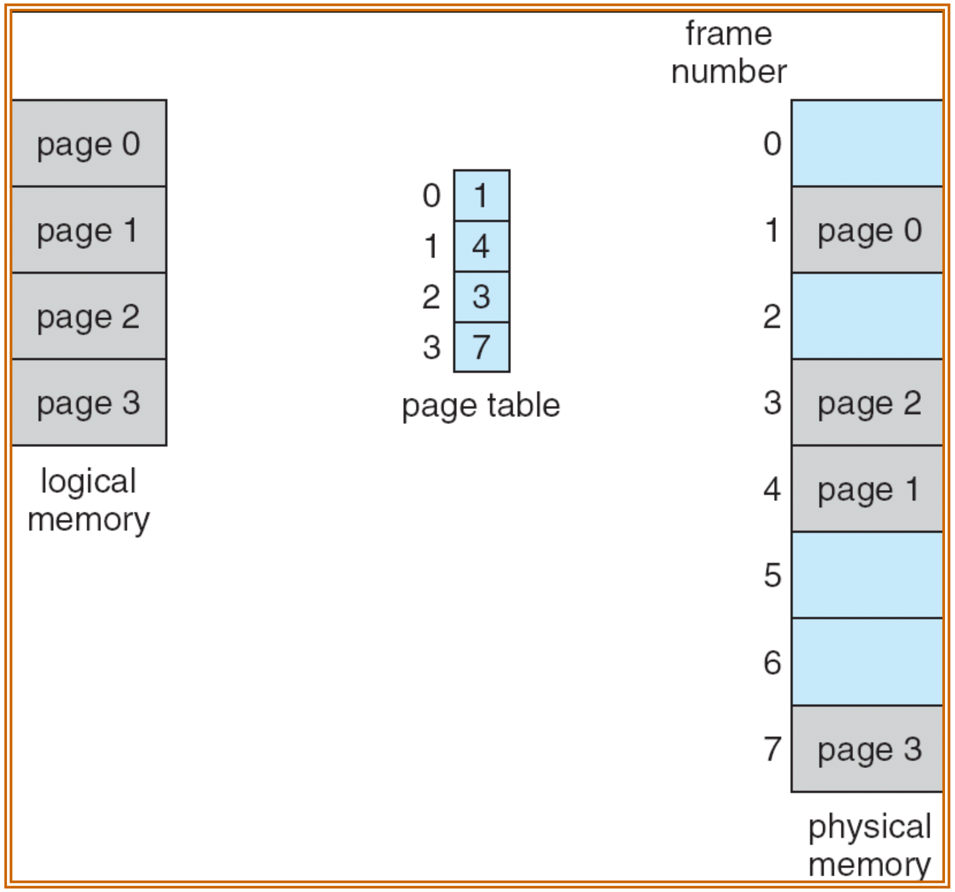
Paging (2/11)

- The programmer/user sees a process as a contiguous block
  - Think of arrays, adjacent instrunctions, …
- What if holes are used totally or partially in physical memory to store a process ?
  - We would need to keep a list of where each of the different logical parts are stored !
  - Too complicated ! Time/Space consuming !

Paging : Basic Scheme (3/11)

1. Break physical memory into fixed-size blocks called **frames**
2. Break logical memory into fixed-size blocks called **pages**
3. Frames and pages are of identical size
4. Address translation is done using a **page table**

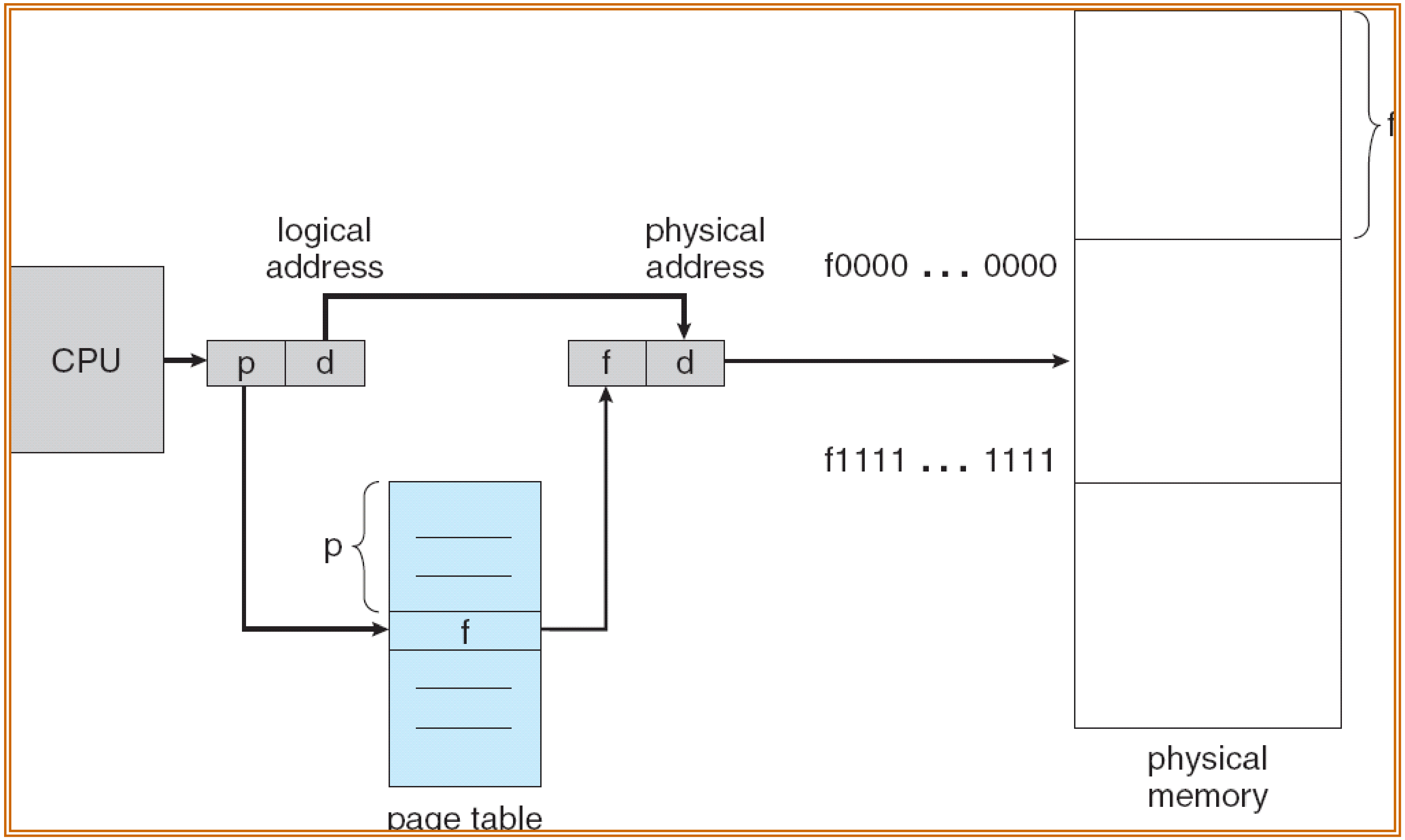
Paging : Basic Scheme (4/11)



Paging : Basic Scheme (5/11)

- Address Translation scheme
  - Address generated by the CPU is divided in 2 parts
    - Page number : used as an index on the page table
    - Offset : address inside the page
  - Every valid entry in the page table holds the address of its corresponding physical frame

Paging : Basic Scheme (6/11)



Paging (7/11)

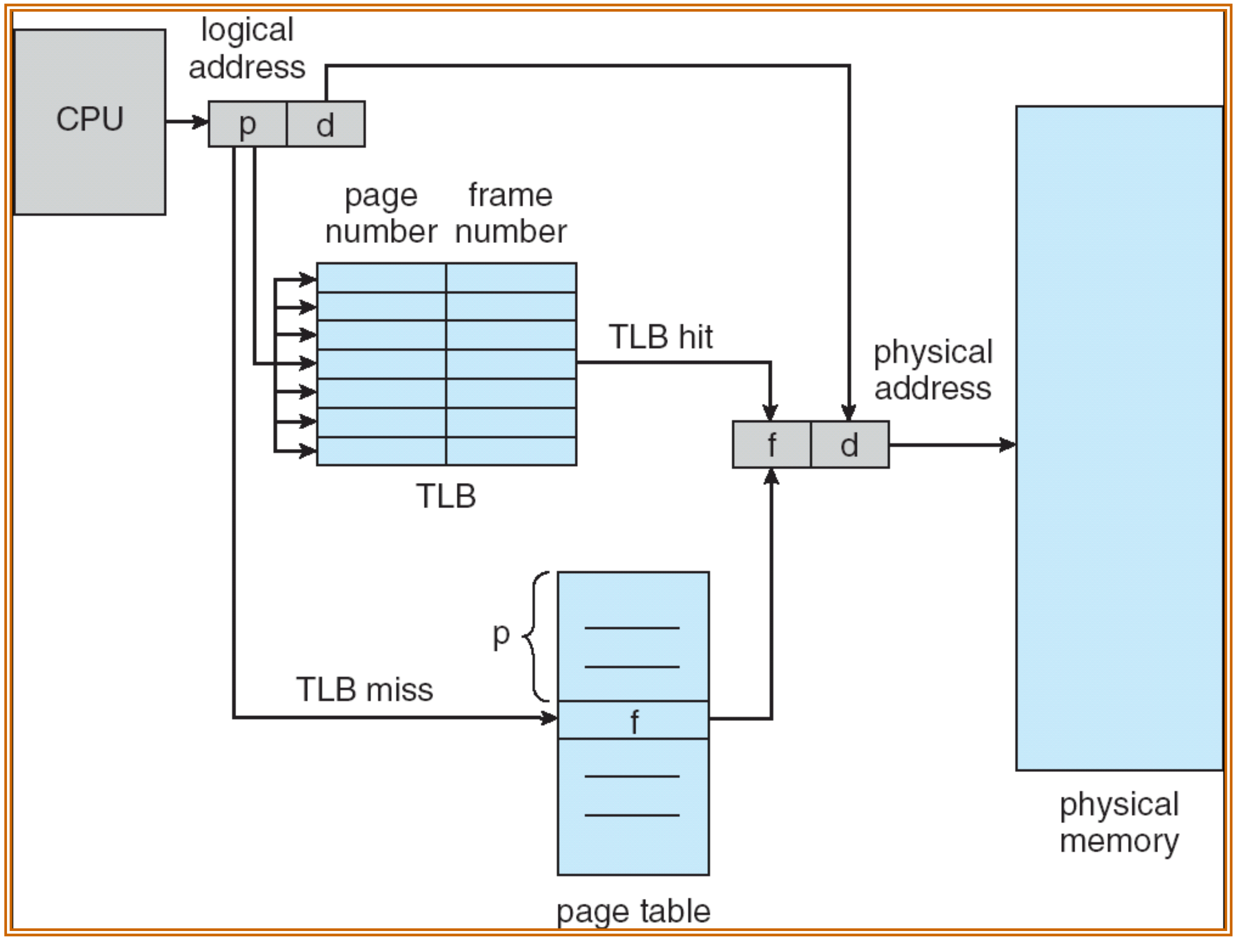
- Advantages
  - Solves the EXTERNAL fragmentation problem
  - Shared pages

- Drawbacks
  - Internal fragmentation : on average, one half page per process
    - Small page sizes are desirable BUT…what about the page table size ?
  - **One more** memory access to the page table in order to produce the physical address => 2 times slower !
    - Address cache : Translation Lookaside Buffer (TLB)

Paging (8/11)

- TLB
  - Associative high-speed memory
  - Its benefits are based on the principles of spatial an temporal locality

Paging with TLB (9/11)



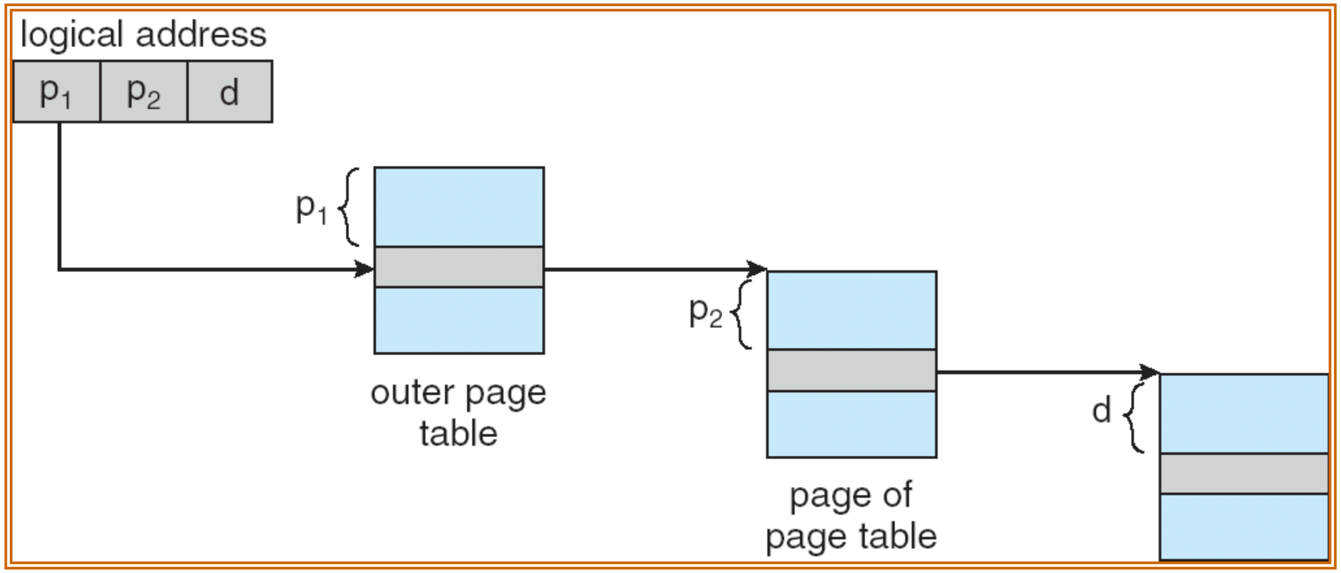
Hierarchical Paging (10/11)

- Large logical address space => large page table size
  - 2^64 with a page size of 4KB => page table of 2^52 entries !

- Use an N-level paging algorithm

Hierarchical Paging (11/11)

- 2-level paging scheme
  - Break the logical address space into Blocks
    - Break blocks into sub-blocks


logical address
p_1
p_2
d
p_1
outer page table
p_2
page of page table
d