Page Table Structure

- Hierarchical Paging
- Hashed Page Tables
- Inverted Page Tables

Hierarchical Page Tables

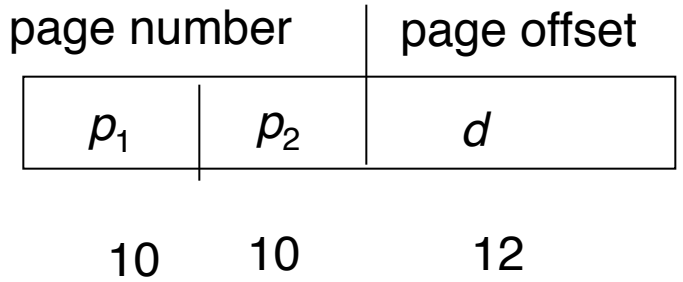
- **Problem**: for very large logical address spaces (2^{32} and 2^{64}) the page table itself becomes very large.
- **Solution**: break up the logical address space into multiple page tables.
- A simple technique is a two-level page table.

Two-Level Paging Example

- A logical address (on 32-bit machine with 4K page size) is divided into:
  - a page number consisting of 20 bits.
  - a page offset consisting of 12 bits.

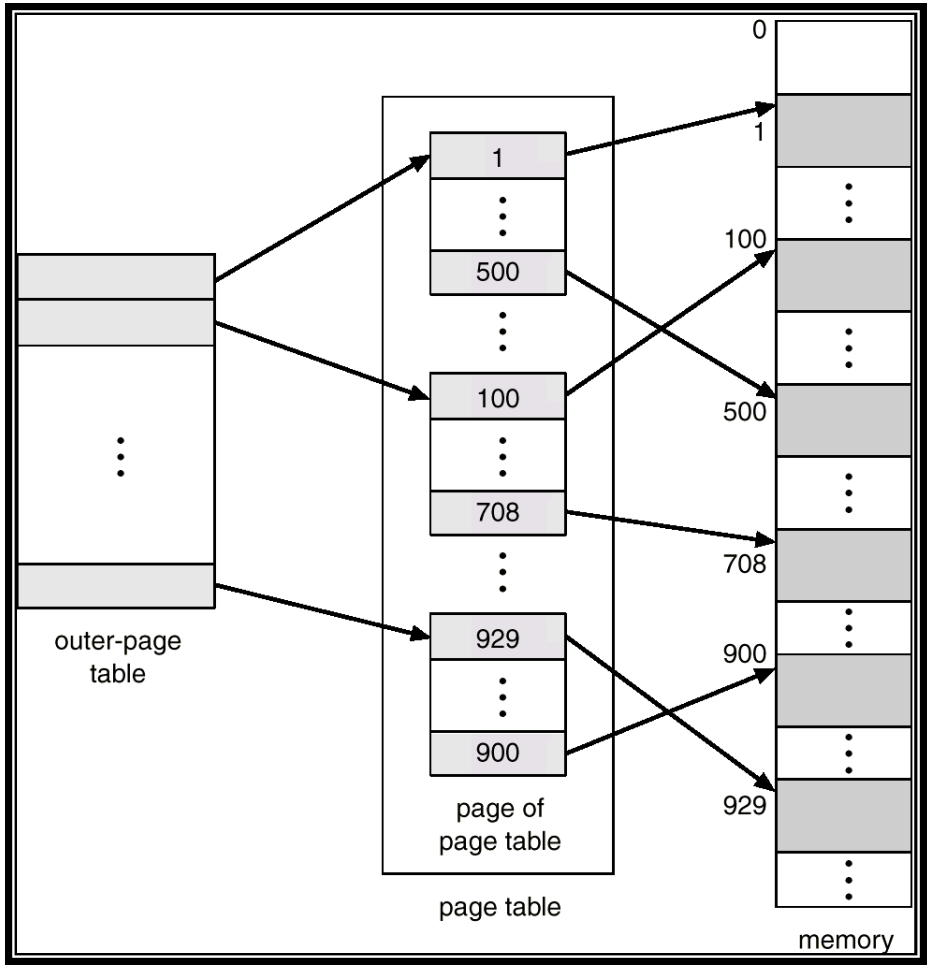
- Since the page table is paged, the page number is further divided into:
  - a 10-bit page number.
  - a 10-bit page offset.

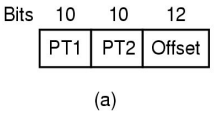
- Thus, a logical address is as follows:

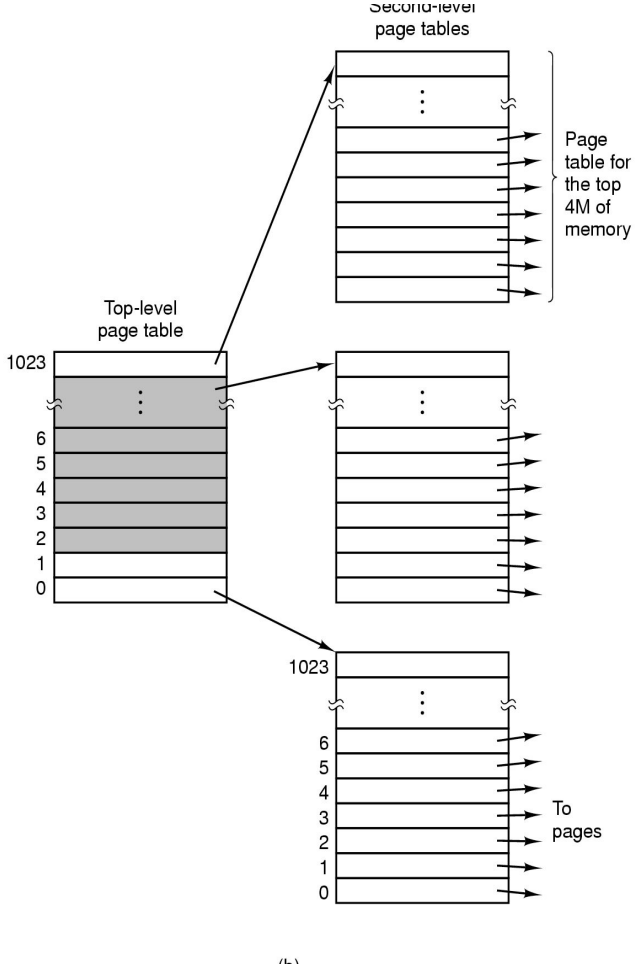


where ***p_1*** is an index into the **outer page table**, and ***p_2*** is the index into the **inner page table.**

Two-Level Page-Table Scheme



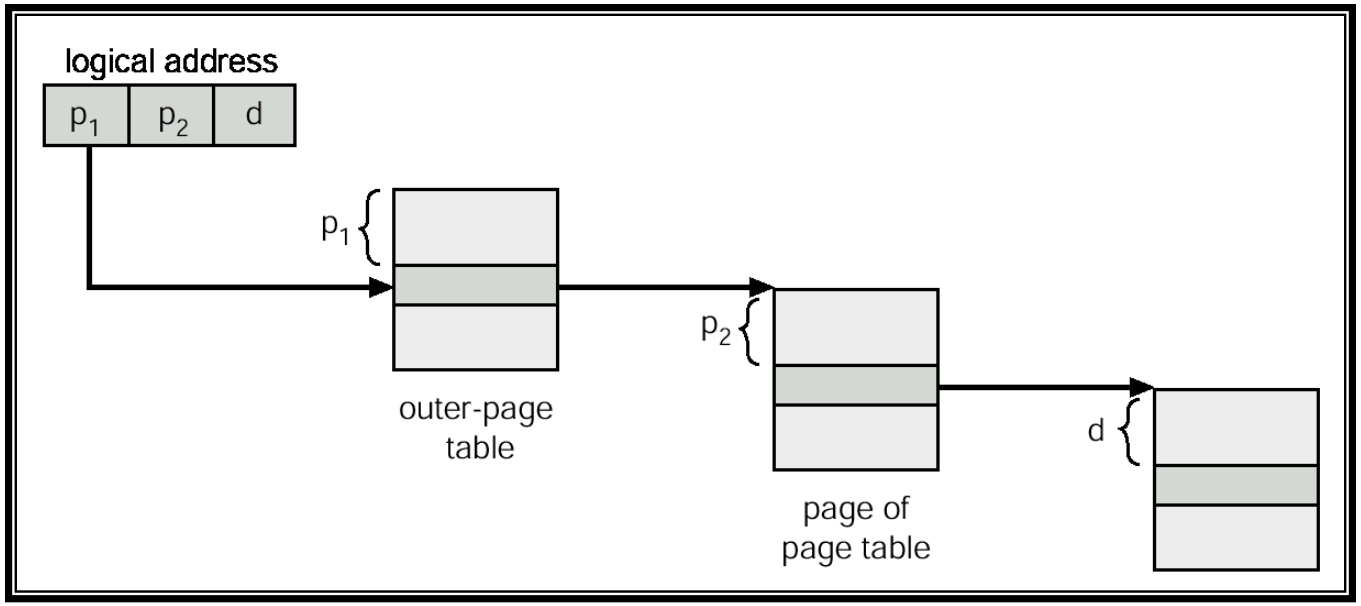




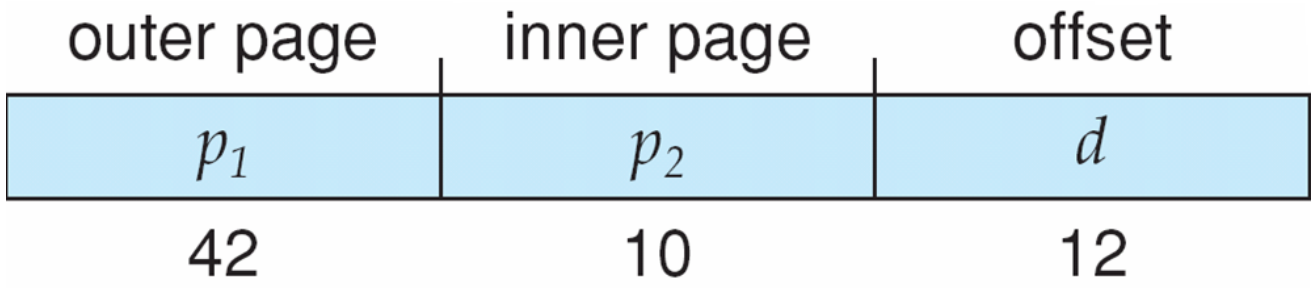
PTBR: base address of the ***outer-page table***
PTLR: number of the ***inner-page tables***

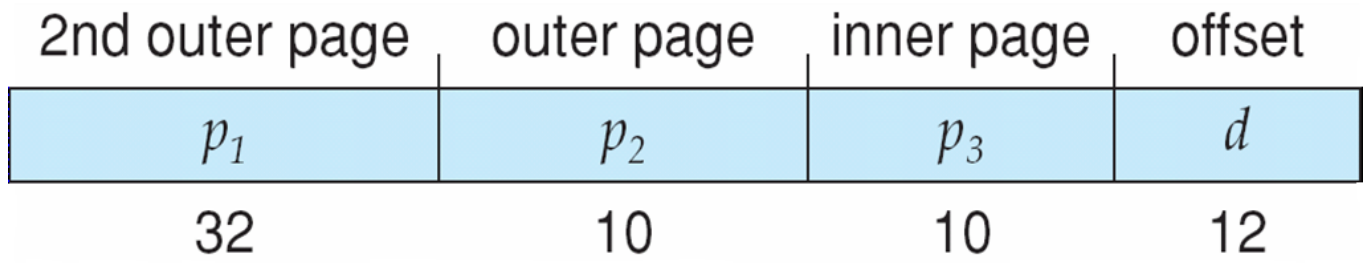
Address-Translation Scheme

- Address-translation scheme for a two-level 32-bit paging architecture



Three-level Paging Scheme for 64-bit address space

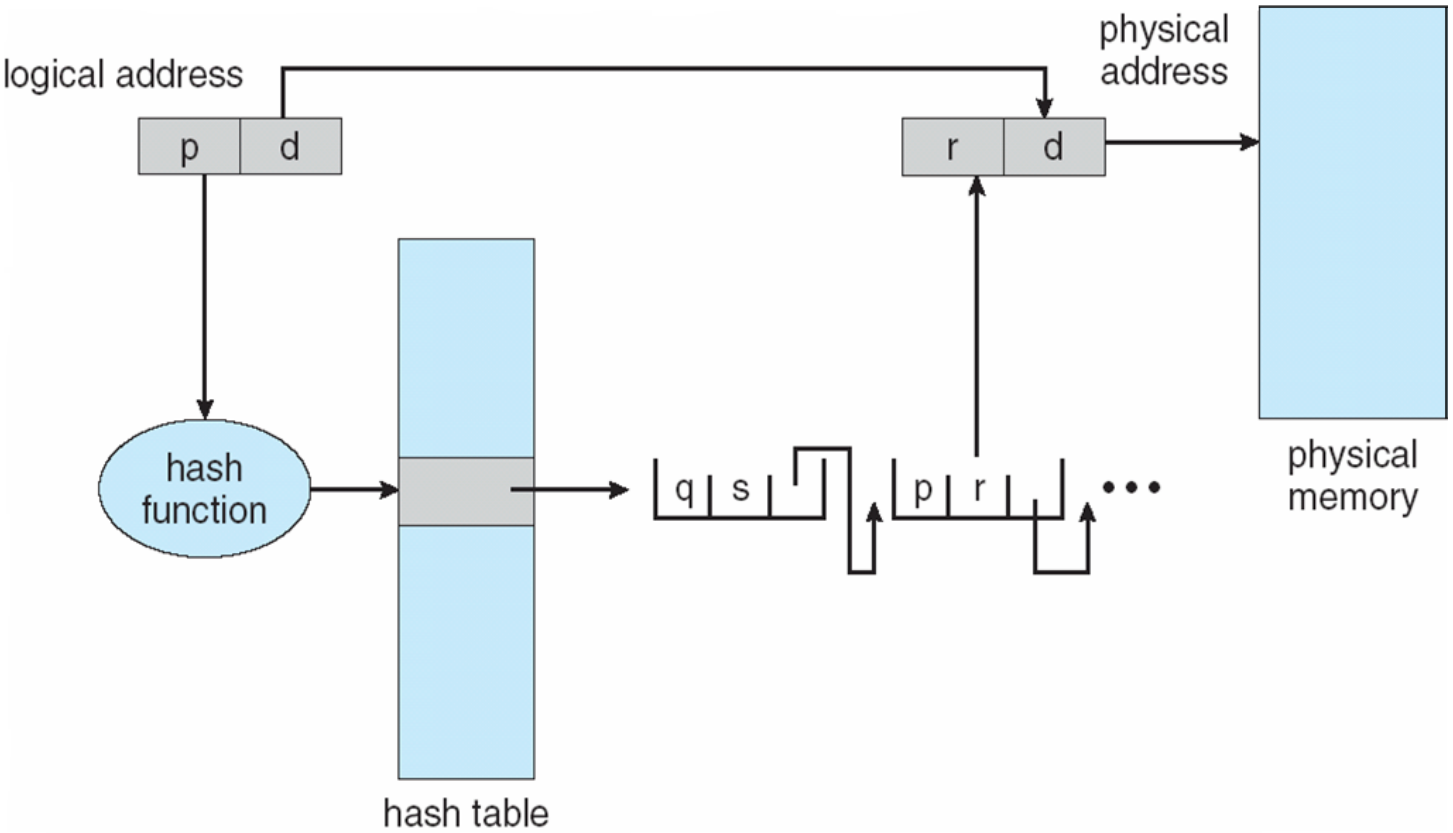




Hashed Page Table Scheme

- Common in address spaces > 32 bits.
- The logical page number is hashed into a page table.
  - This page table contains a chain of elements in the form of <page#, frame#> hashing to the same location.
- Logical page number is compared in this chain searching for a match.
  - If a match is found, the corresponding physical frame# is extracted.

Hashed Page Table



Hash tables

Linar Probing

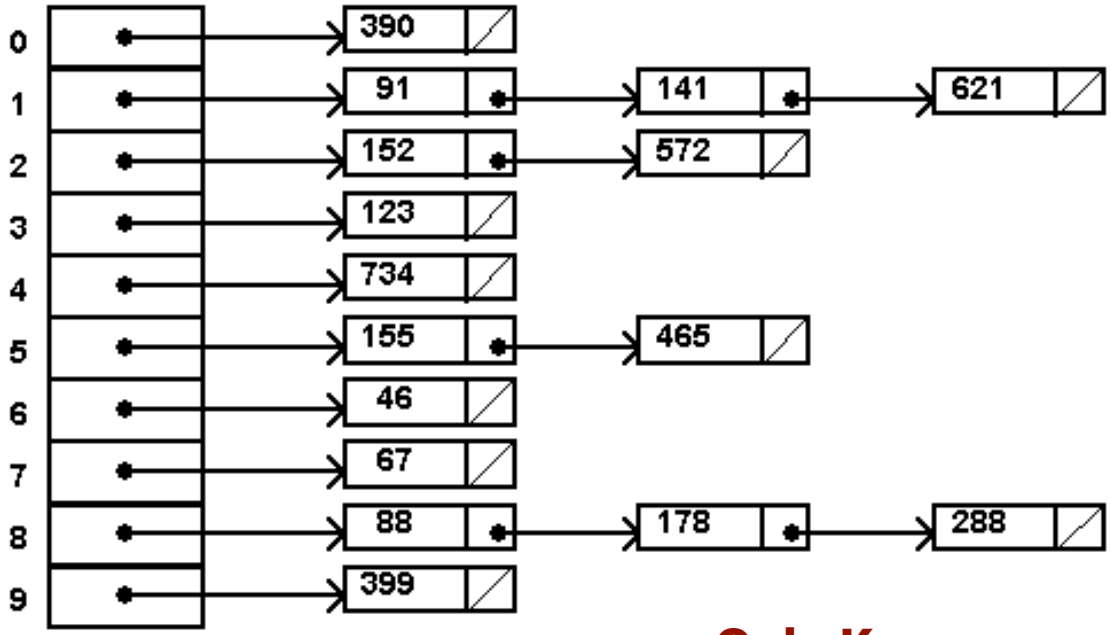
0	
1	141
2	621
3	123
4	
5	465
6	46
7	67
8	88
9	288
10	390
11	91
12	152
13	572
14	734
15	155
16	
17	
18	178
19	399

67 % 20 = 7
46 % 20 = 6
88 % 20 = 8
91 % 20 = 11

...

734 % 20 = 14

Chained Overflow



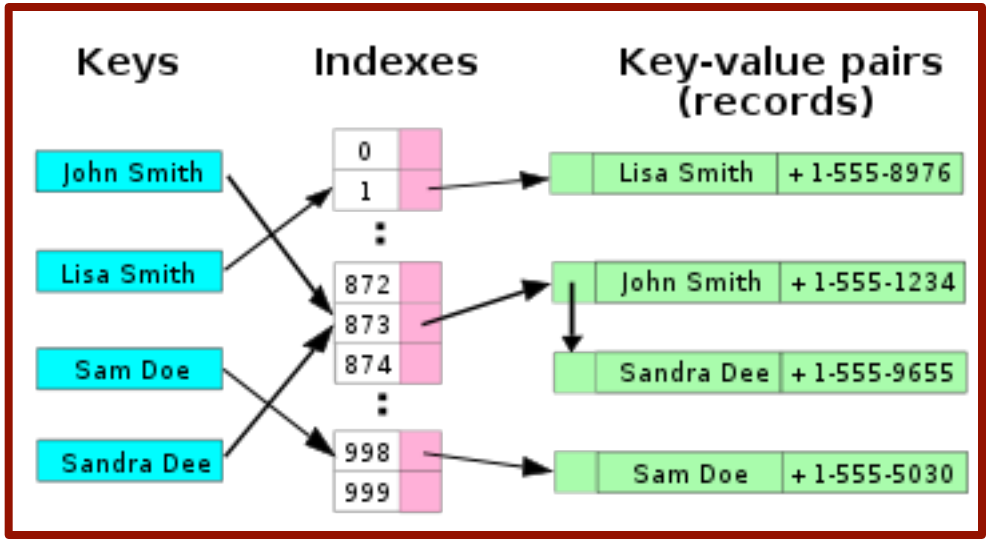
Hash Function

67 % 10 = 7
46 % 10 = 6
88 % 10 = 8
91 % 10 = 1
...
734 % 10 = 4

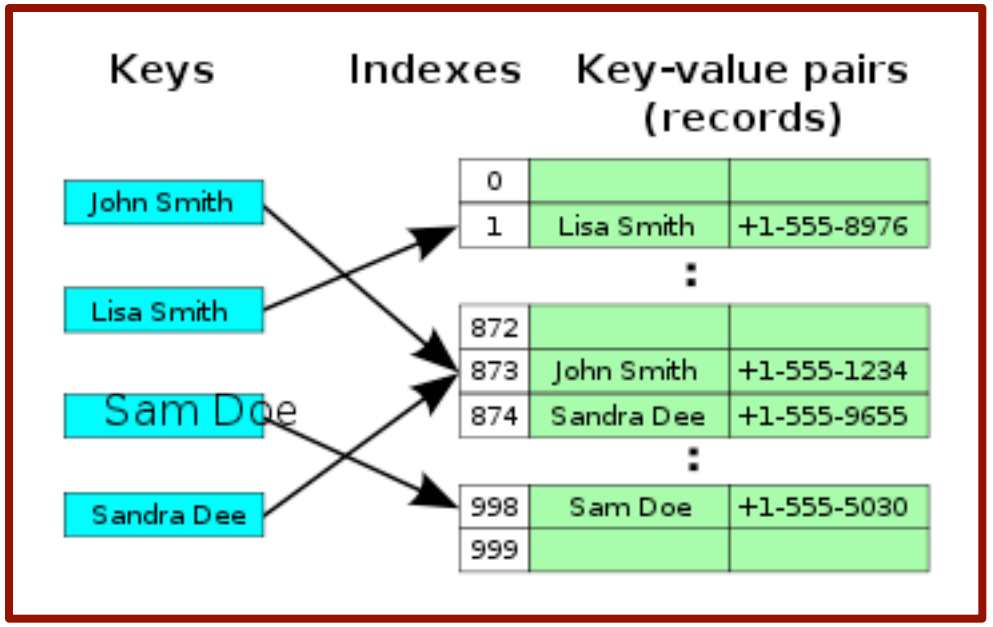
Only Key,
Value not shown.
(Key, Value) pairs.

Hash Table Example of Telephone Directory

Hash collision resolved by chaining



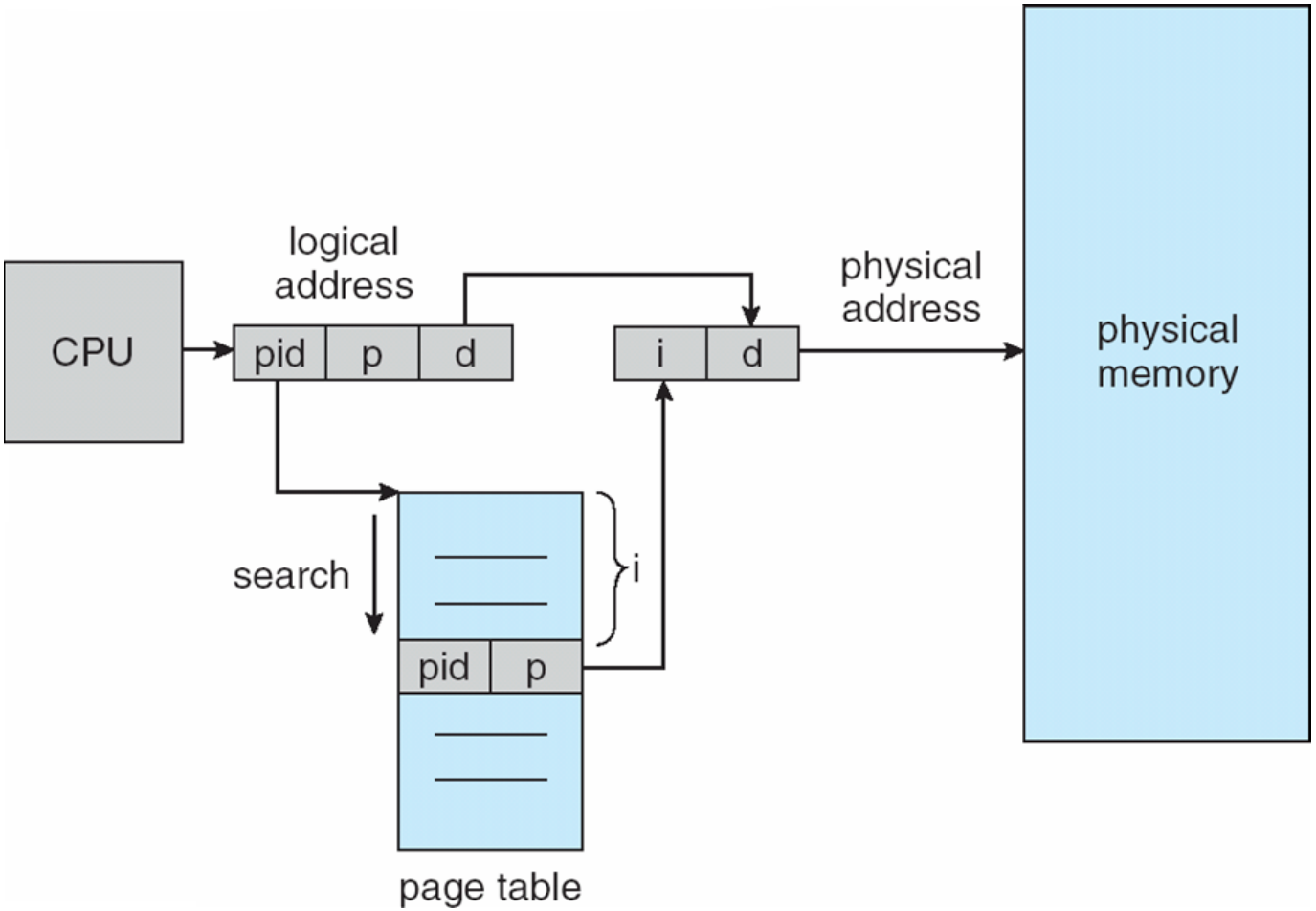
Hash collision resolved by linear probing (interval = 1)



Inverted Page Table

- One entry for each frame of physical memory.
- Entry consists of the logical address of the page stored in that real memory location, with information about the process that owns that page.
- Decreases memory needed to store each page table, but increases time needed to search the table when a page reference occurs.
- Use hash table to limit the search to one — or at most a few — page-table entries.

Inverted Page Table Architecture

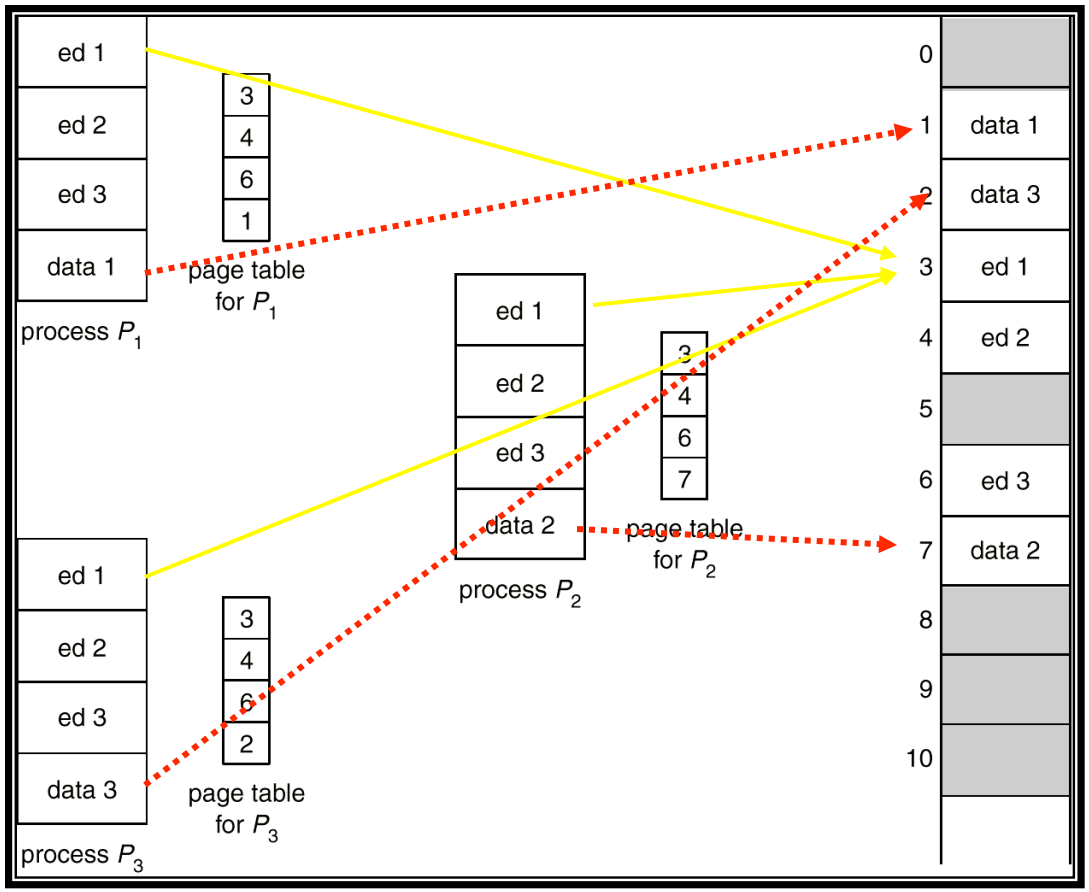


Shared Pages

- Shared code
  - One copy of read-only (reentrant) code shared among processes (i.e., text editors, compilers, window systems).
  - Shared code must appear in same location in the logical address space of all processes

- Private code and data
  - Each process keeps a separate copy of the code and data.
  - The pages for the private code and data can appear anywhere in the logical address space.

Shared Pages Example

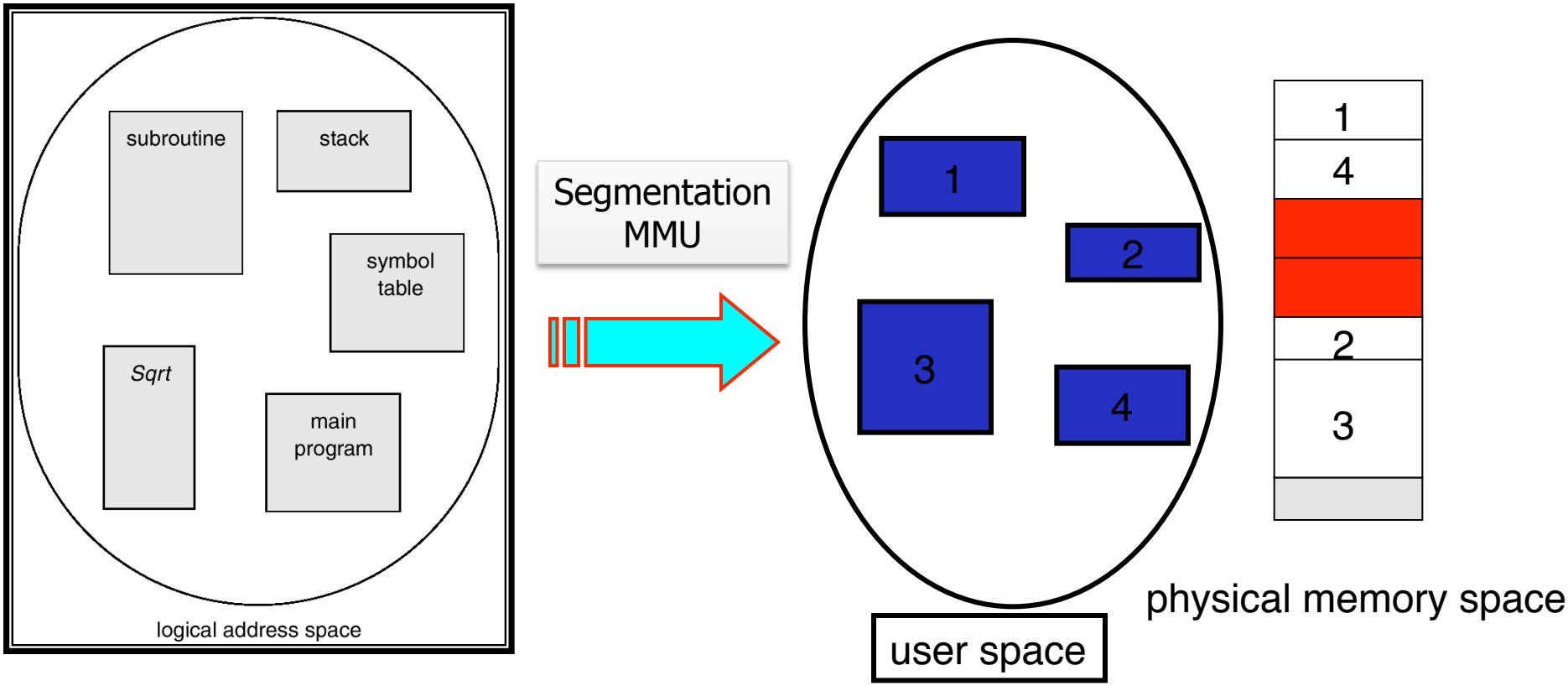


Segmentation

- Memory-management scheme that supports user view of program.
- User does not look at program as an array of memory data
- A program is a collection of segments.
  - A segment is a logical unit such as:
    main program,
    procedure,
    function,
    method,
    object,
    local variables, global variables,
    common block,
    stack,
    symbol table,
    arrays

User's View of a Program

& Logical View of Segmentation



Segmentation Architecture

- Logical address consists of a two tuple:
  <segment-number, offset>,

- *Segment table* – maps two-dimensional logical addresses; each table entry has:
  - base – contains the starting physical address where the segments reside in memory.
  - *limit* – specifies the length of the segment.

- *Segment-table base register (STBR)* points to the segment table's location in memory.

- *Segment-table length register (STLR)* indicates number of segments used by a program;
  segment number *s* is legal if *s* < STLR.

Segmentation Architecture (Cont.)

Features:

- Relocation.
  - dynamic
  - by segment table
- Sharing.
  - shared segments
  - same segment number
- Allocation.
  - first fit/best fit
  - external fragmentation

Segmentation Architecture (Cont.)

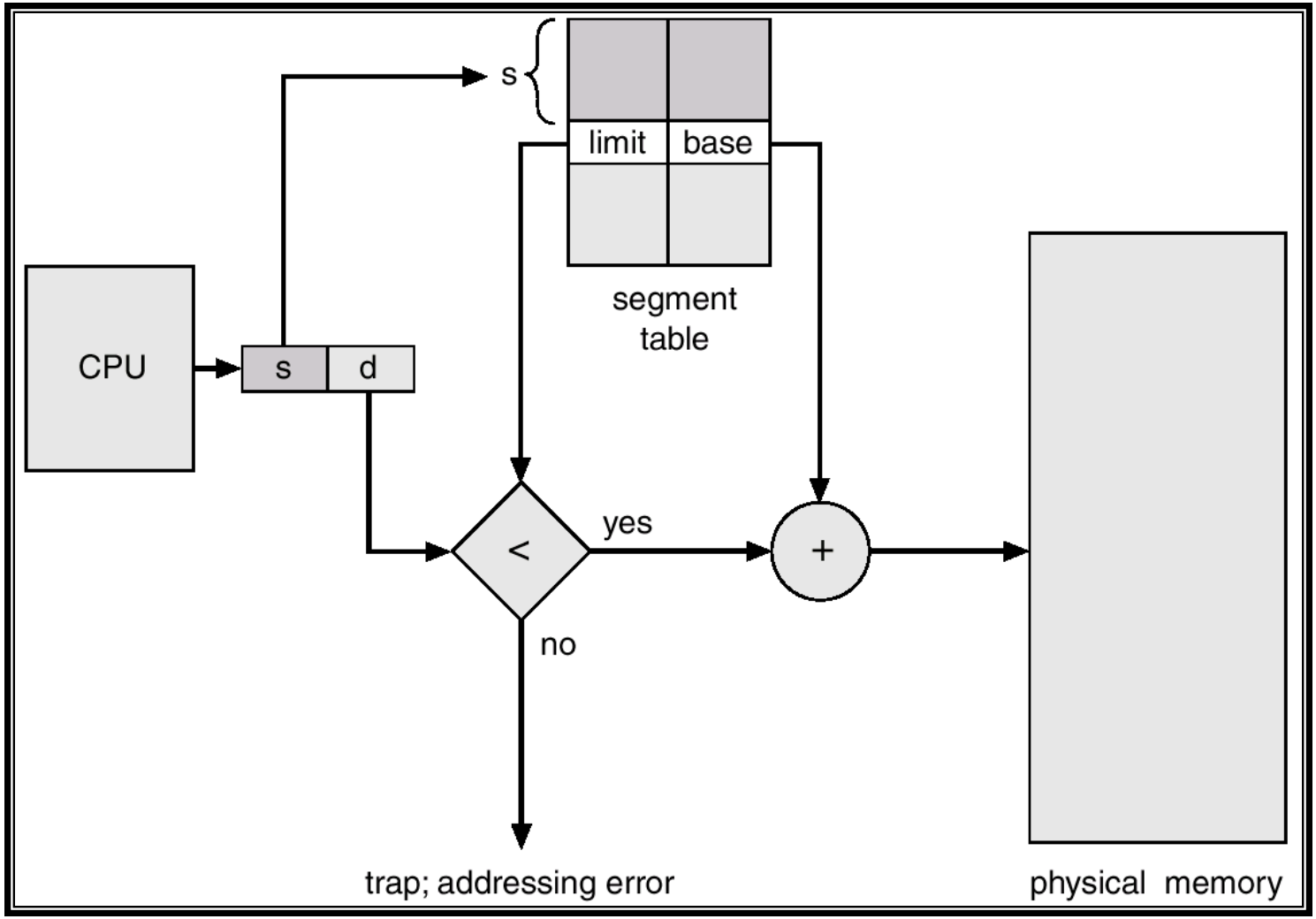
- Protection. With each entry in segment table associate:
  - validation bit = 0 ⇒ illegal segment
  - read/write/execute privileges

- Protection bits associated with segments; code sharing occurs at segment level.

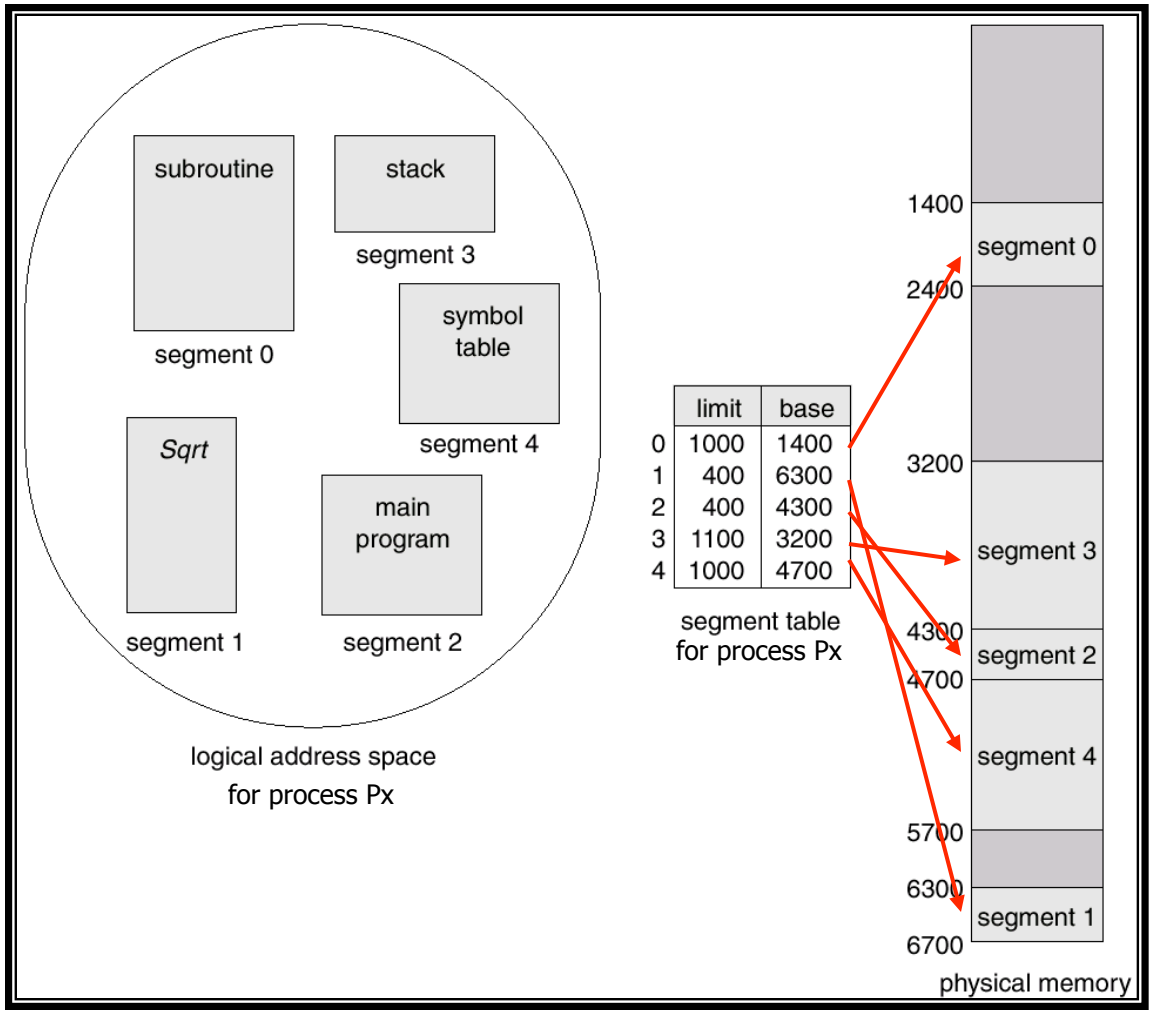
- Since segments vary in length, memory allocation is a dynamic storage-allocation problem.

- A segmentation example is shown in the following diagram

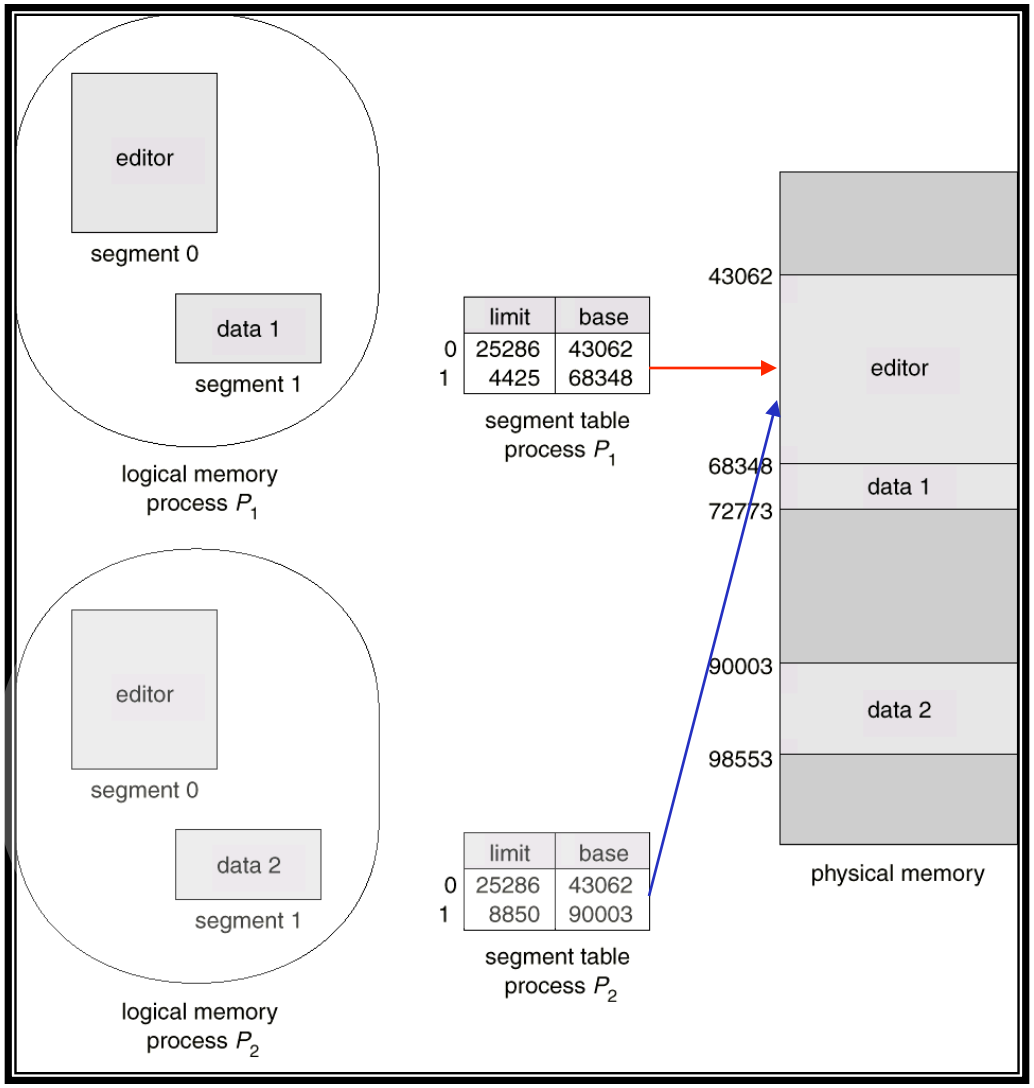
Segmentation Hardware

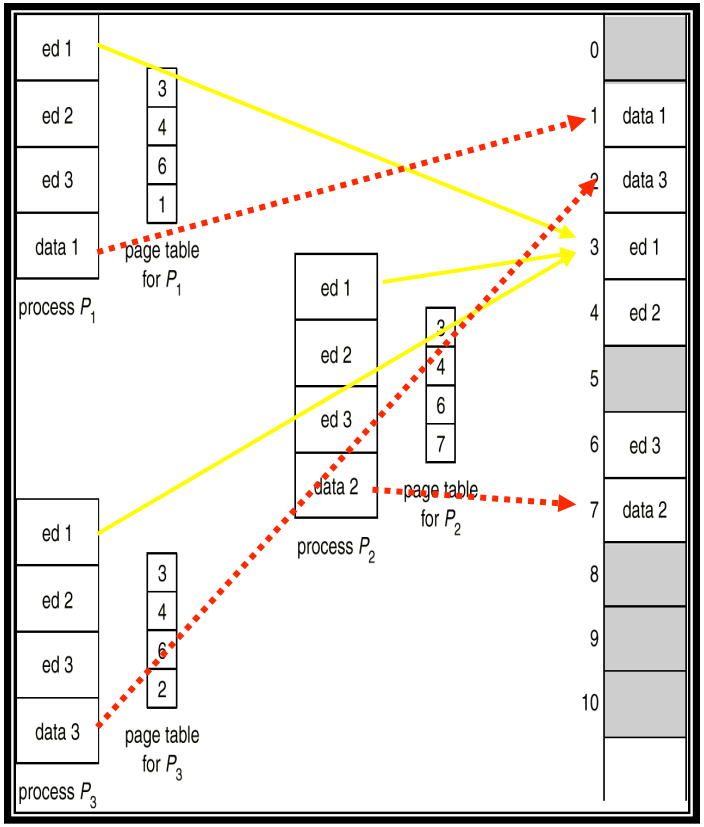


Example of Segmentation



Sharing of Segments





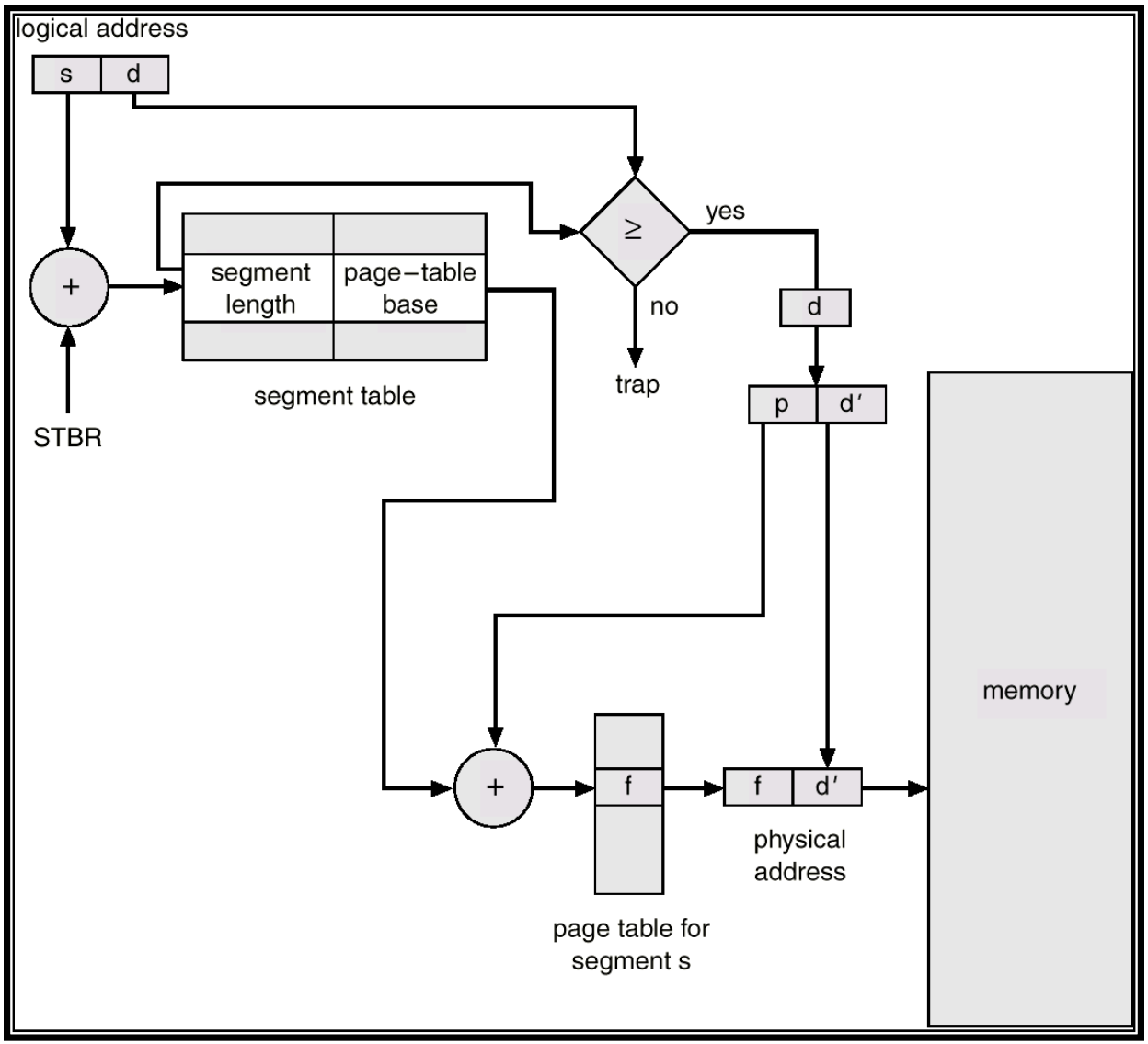
Segmentation with Paging

- The MULTICS and Intel 80386 systems solved problems of external fragmentation and lengthy search time for empty slot by paging the segments.

- Solution differs from pure segmentation in that the segment-table entry contains not the base address of the segment, but rather the base address of a *page table* for this segment.

PowerPc Processor Memory Management:
http://playground.sun.com/1275/bindings/ppc/release/ppc-2_1.html#HDR4

MULTICS Address Translation Scheme



Intel 80386

- The Intel 386 uses segmentation with paging for memory management with a two-level paging scheme.
- Logical address consists of a segment selector (in one of the segment registers of CPU) and 32 bits address that is generated on the address bus.
- Each process can have up to 8000 private segments and up to 8000 shared segments to be shared by other processes.
- Each segment can have up to 1 million pages (each page is 4K bytes) hence the size of a segment can be up to 4 Giga bytes.
- The pages of a segment are scattered throughout the memory frames.
- A 32 bit address points to the start of a segment of pages and a 32 bit address from address bus is used as the offset in that segment. This offset is used to access the locations in the pages of the segment.
- Intel 386 has hardware to support virtual memory management effectively.

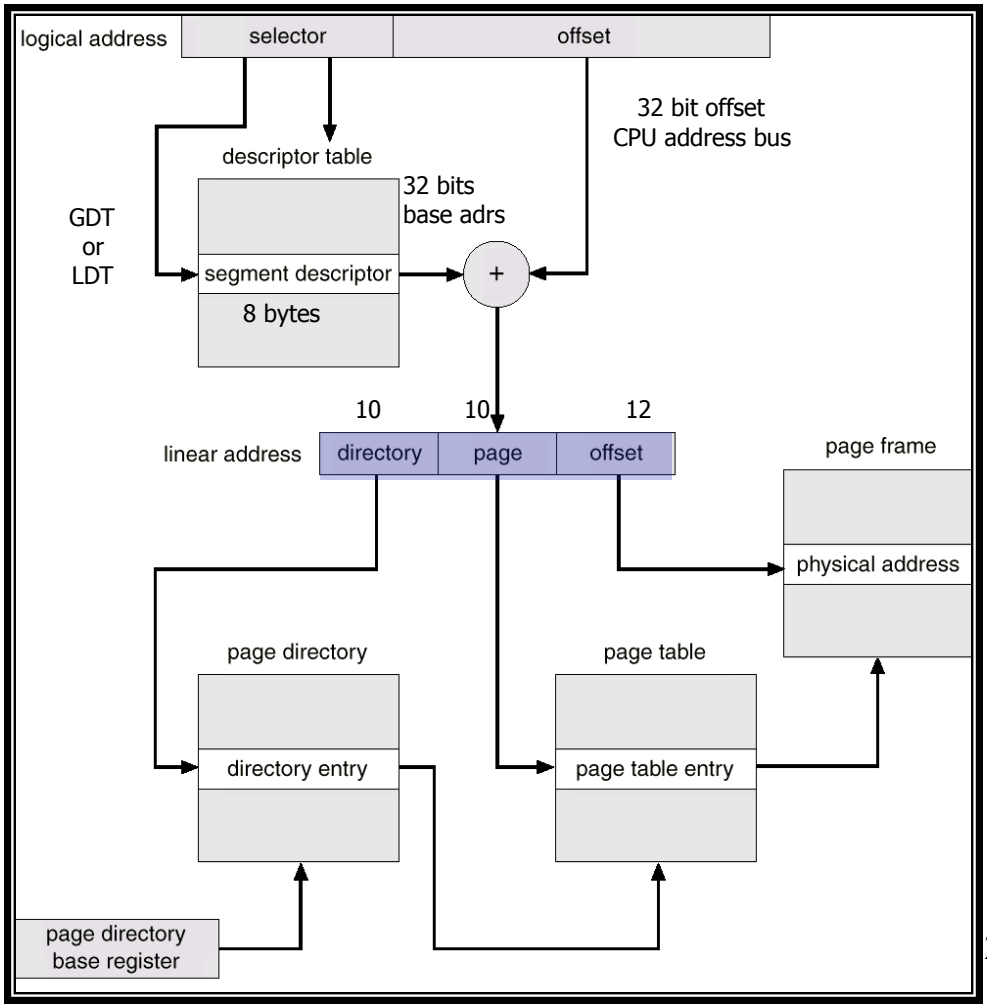
Segmentation with Paging – Intel 386

- The Intel 386 uses segmentation with paging for memory management with a two-level paging scheme.

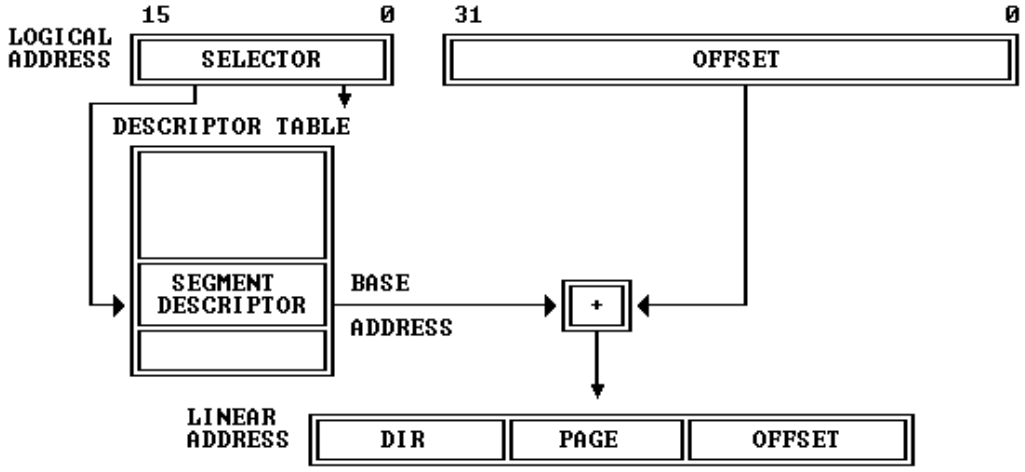
Intel 80386
address translation

Selector part of logical address.

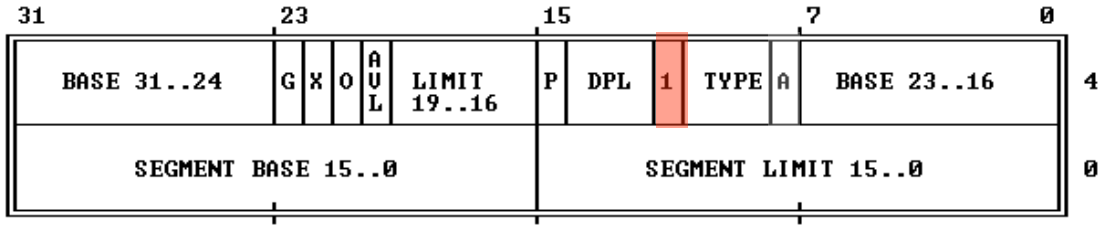
Stored in one of the CPU's segment registers:
CS, DS, SS, ES, ..



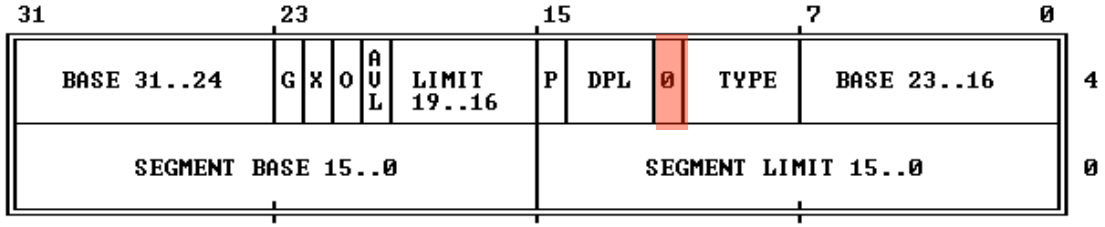
80386 Architecture for MMU



DESCRIPTORS USED FOR APPLICATIONS CODE AND DATA SEGMENTS



DESCRIPTORS USED FOR SPECIAL SYSTEM SEGMENTS



A – ACCESSED
AVL – AVAILABLE FOR USE BY SYSTEMS PROGRAMMERS
DPL – DESCRIPTOR PRIVILEGE LEVEL
G – GRANULARITY
P – SEGMENT PRESENT

80386 Architecture

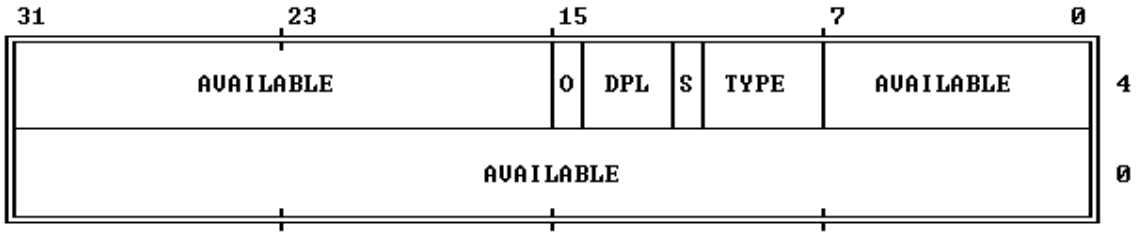
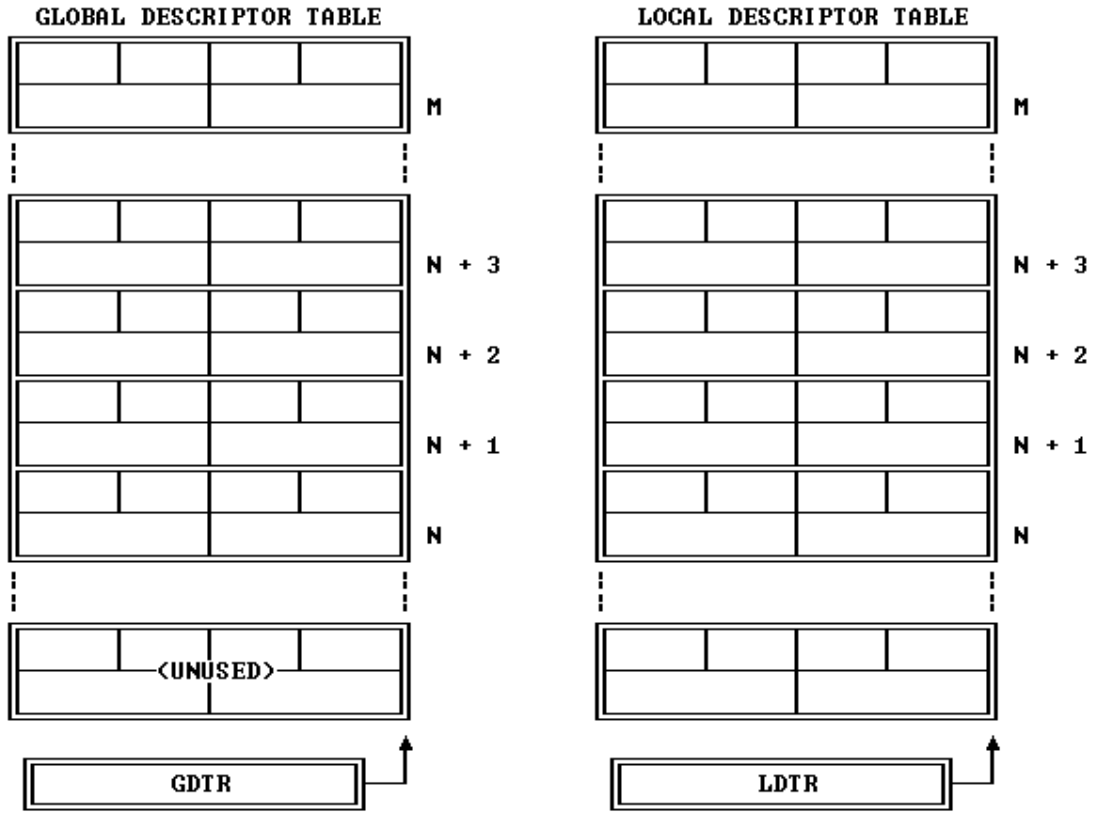


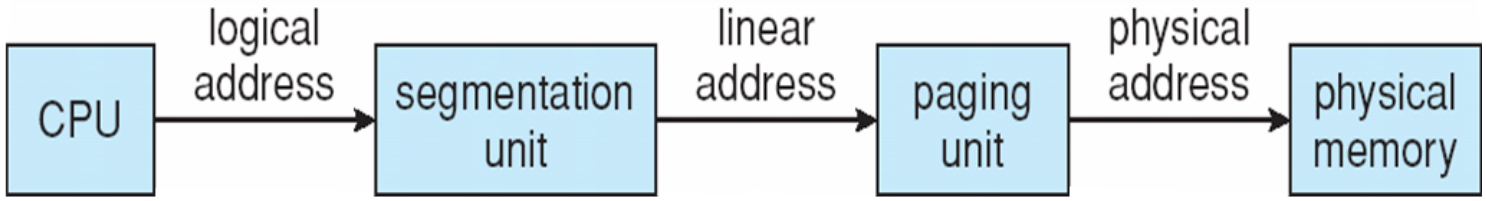
Figure 5-5. Descriptor Tables

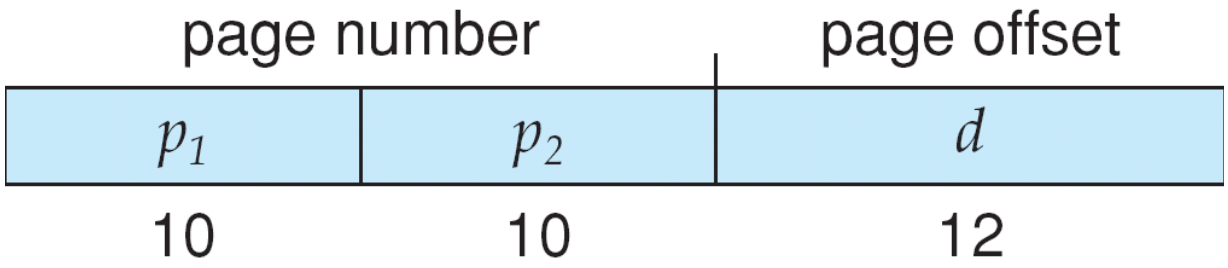


Example: The Intel Pentium

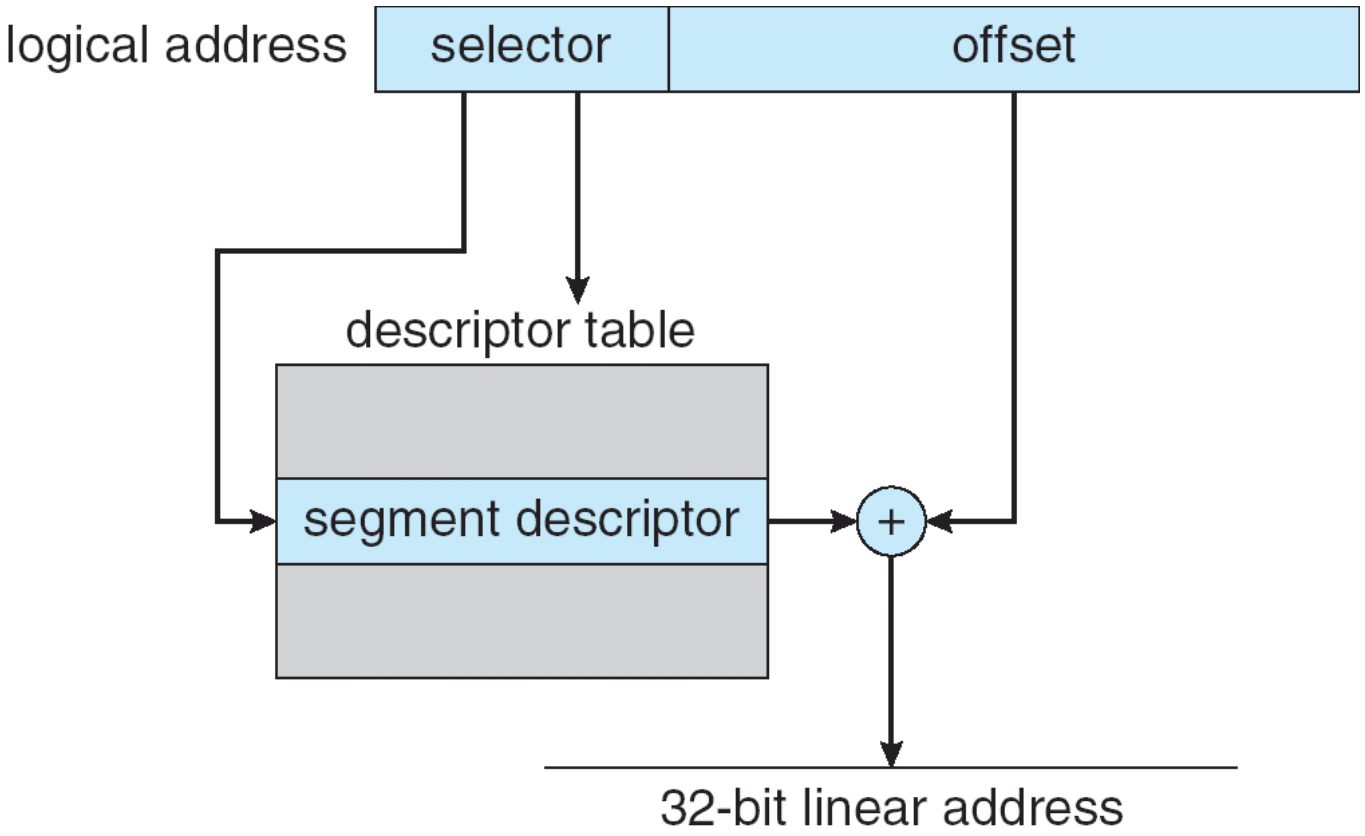
- Supports both segmentation and segmentation with paging
- CPU generates logical address
  - Given to segmentation unit
    - Which produces linear addresses
  - Linear address given to paging unit
    - Which generates physical address in main memory
    - Paging units form equivalent of MMU

Logical to Physical Address Translation in Pentium

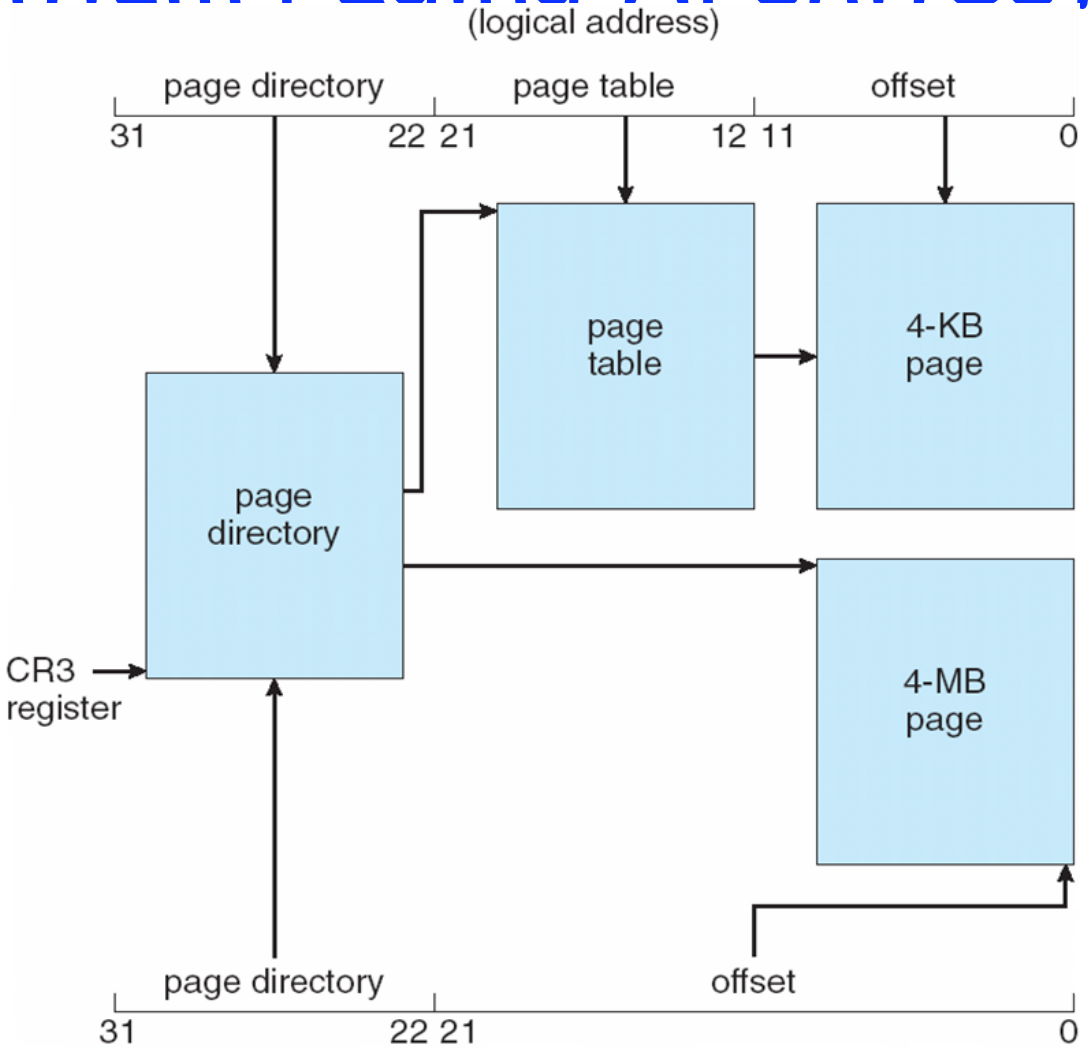




Intel Pentium Segmentation

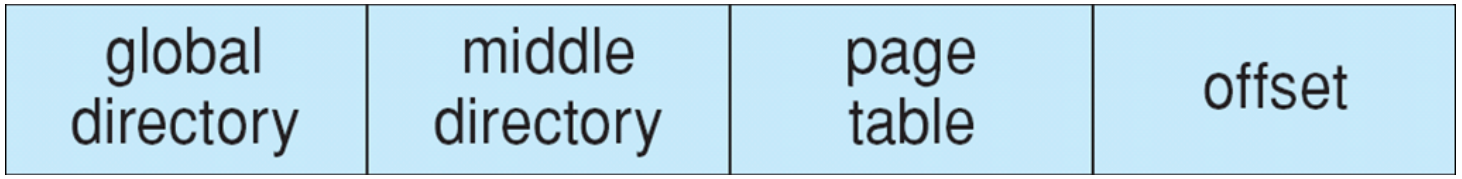


Pentium Paaina Architecture



Linear Address in Linux

Broken into four parts:



Three-level Paging in Linux