Memory management

Linux Memory Management

Contents

- Module Coverage
  - Virtual Memory
  - Memory Addressing
  - Paging in Hardware
  - Memory Area Management
  - Memory Management functions

Memory Management Basics

- Bit
  - Bit is a unit of memory
- Word
  - Smallest addressable unit of processor
- MMU(Memory Management Unit)
  - It is the hardware that manages memory and performs virtual to physical address translations
- Page
  - Basic unit of memory management

Background

- Program must be brought into memory and placed within a process for it to be run.
- Input queue – collection of processes on the disk that are waiting to be brought into memory to run the program.

Virtual Memory

- Virtual Memory
  - Technique that allows the execution of processes that may not be completely in memory.
  - Programs can be larger than physical memory
- Goals
  - Allow virtual address spaces that are larger than the physical address space
- Methods
    - Allow pages from the virtual address space to be stored in secondary memory, as well as primary memory
    - Move pages between secondary and primary memory so that they are in primary memory when they are needed

- Demand Paging
- Paging is a memory allocation strategy by transferring a fixed-sized unit of the virtual address space called virtual page whenever the page is needed to execute a program. As the size of frames and pages are the same, any logical page can be placed in any physical frame of memory.
- Every processes will be logical divided and allocate in the virtual address space. There is a page table in the virtual memory to allocate and keep tracking of the pages to map into the frames.
- Demand paging decreases the paging time and physical memory needed because only the needed pages will be paged and the reading time of the unused pages can be avoided.

- Page Fault Problem
- A page fault occurs when a program try to use a page that is not in the memory, due to demand paging will only paged the pages into the memory when it is needed. For example in figure 1.9, if the program try to use Page 1 for Process A in memory, the operating system will interrupt occurs as a result of trying access a missing page because Page 1 is not paged in the memory.

- Problem- No Free Frames
- When all the frames in the memory is been used, the other problem will occurs. This will cause the pages is unable to paged into the memory.

- Solution- Page Replacement Algorithms

- Solution for no free frames problem is to
- find a memory frame that is idle and free
- the frame using a page replacement algorithm.
- There are three common types of page replacement algorithm
- such as First in First out (FIFO)
- Optimal
- Least Recently Used (LRU).

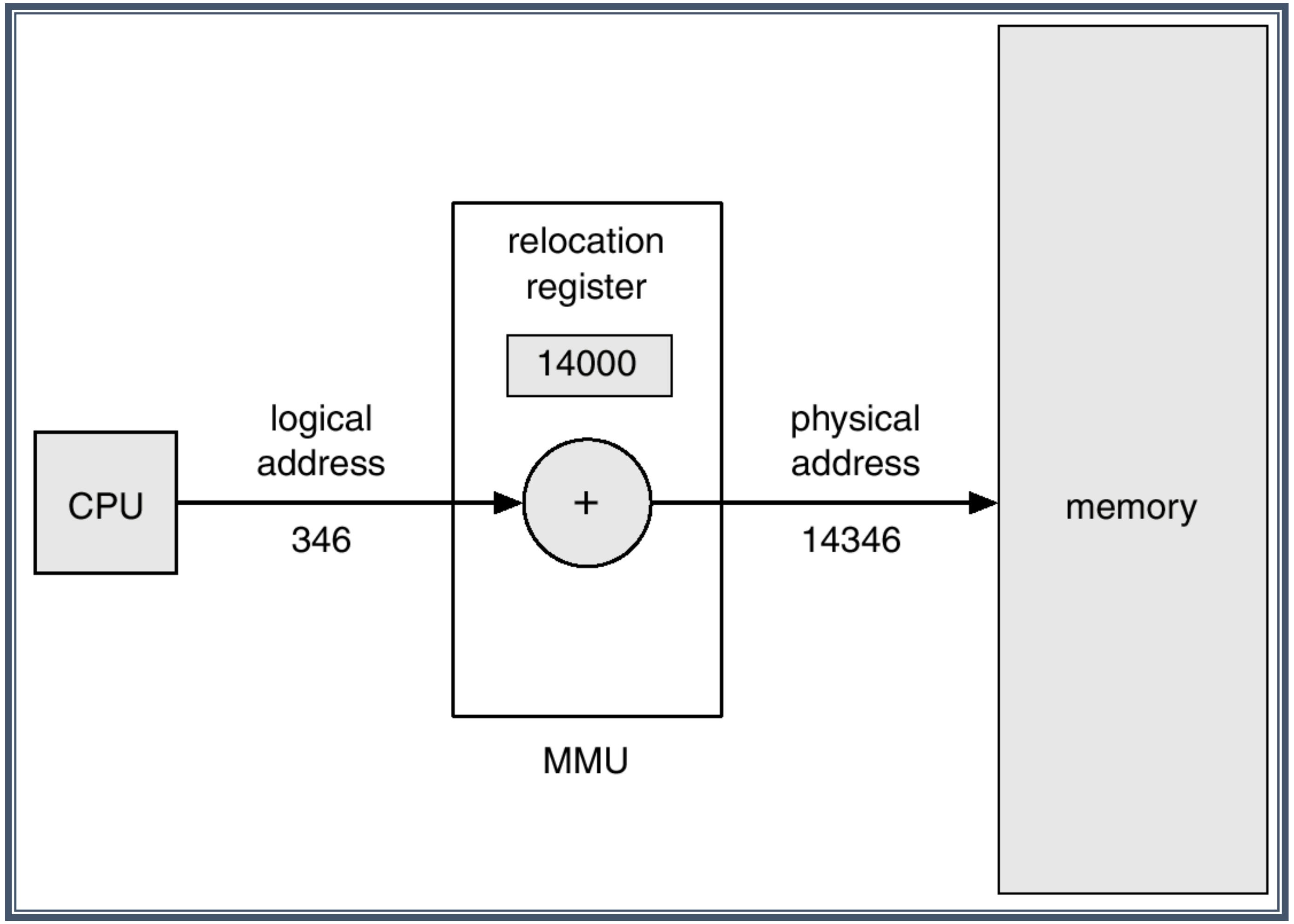
Logical vs. Physical Address Space

- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.
  - *Logical address* – generated by the CPU; also referred to as *virtual address*.
  - *Physical address* – address seen by the memory unit.
- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme.

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address.
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory.
- The user program deals with *logical* addresses; it never sees the *real* physical addresses.

Dynamic relocation using a relocation register



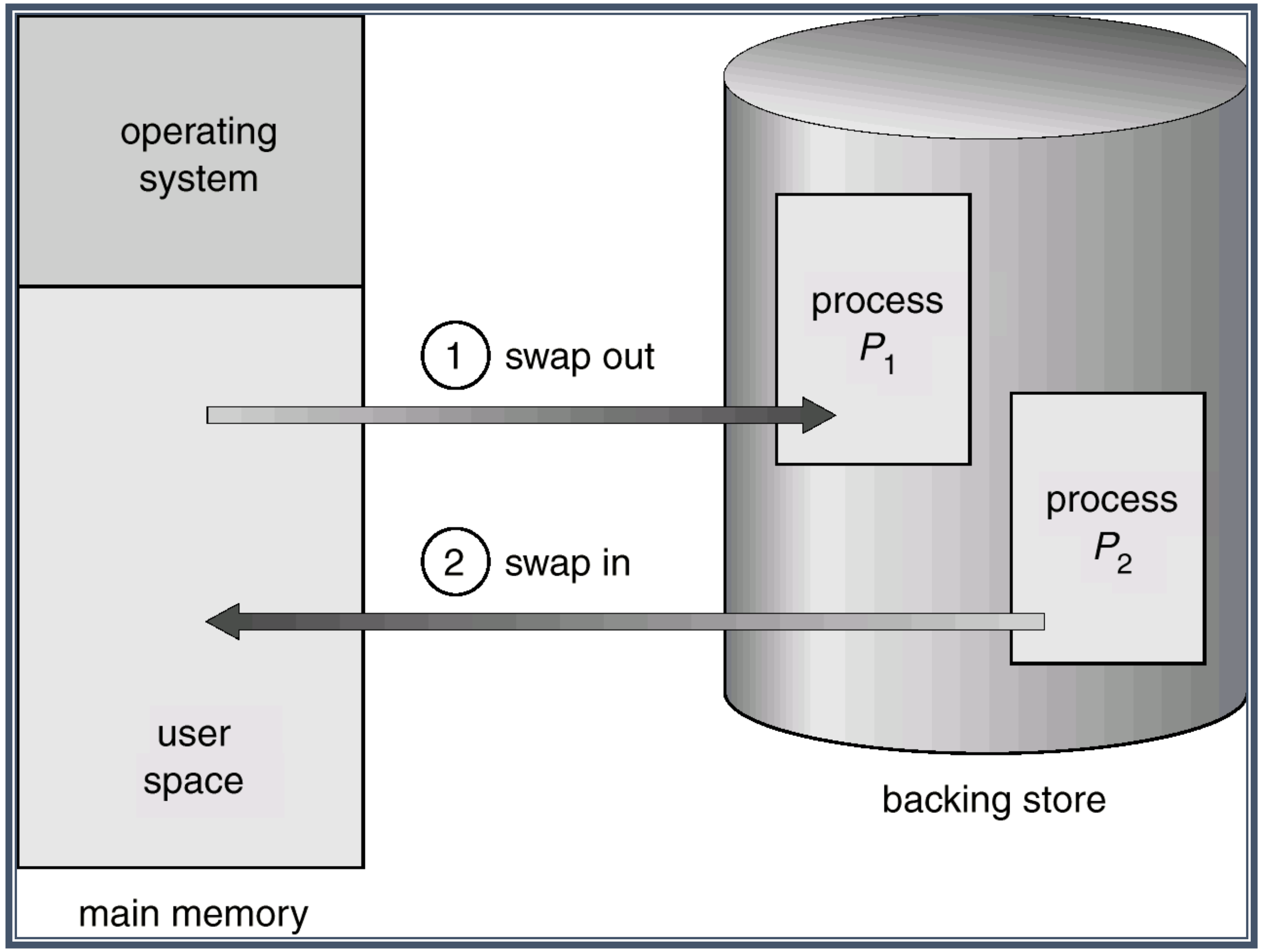
Dynamic Loading

- Routine is not loaded until it is called
- Better memory-space utilization; unused routine is never loaded.
- Useful when large amounts of code are needed to handle infrequently occurring cases.
- No special support from the operating system is required implemented through program design.

Swapping

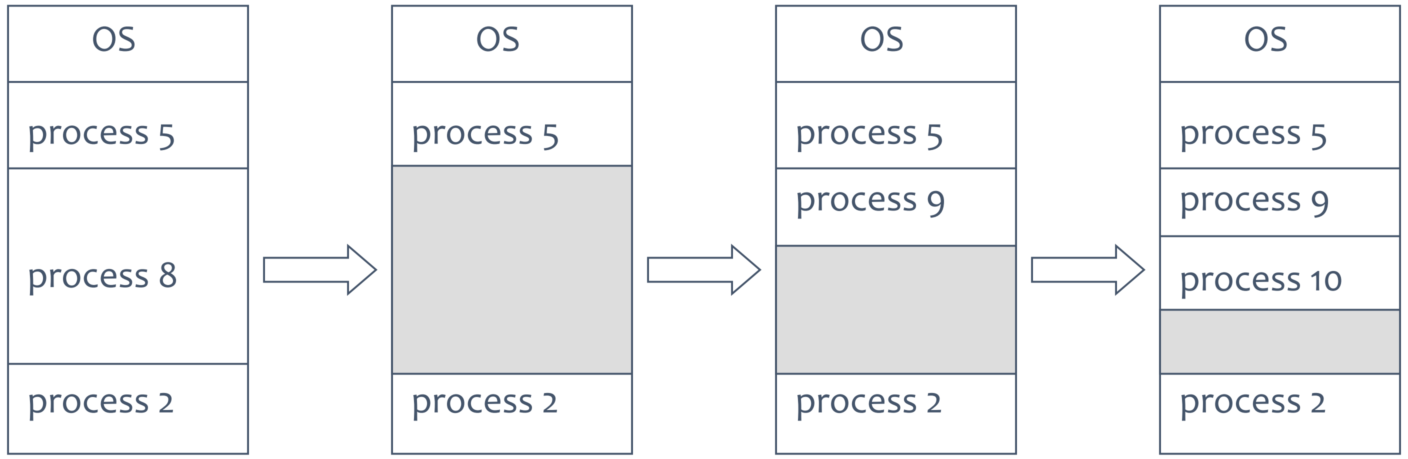
- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution.
- Backing store – fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images.
- Roll out, roll in – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed.
- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped.

Schematic View of Swapping



Contiguous Allocation

- 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)



Dynamic Storage-Allocation Problem

- How to satisfy a request of size n from a list of free holes.
- First-fit: Allocate the *first* hole that is big enough.
- Best-fit: Allocate the *smallest* hole that is big enough; must search entire list, unless ordered by size. Produces the smallest leftover hole.
- Worst-fit: Allocate the *largest* hole; must also search entire list. Produces the largest leftover hole.
- First-fit and best-fit better than worst-fit in terms of speed and storage utilization.

Fragmentation

- External Fragmentation – total memory space exists to satisfy a request, but it is not contiguous.
- Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used.
- Reduce external fragmentation by compaction
    - Shuffle memory contents to place all free memory together in one large block.
    - Compaction is possible *only* if relocation is dynamic, and is done at execution time.

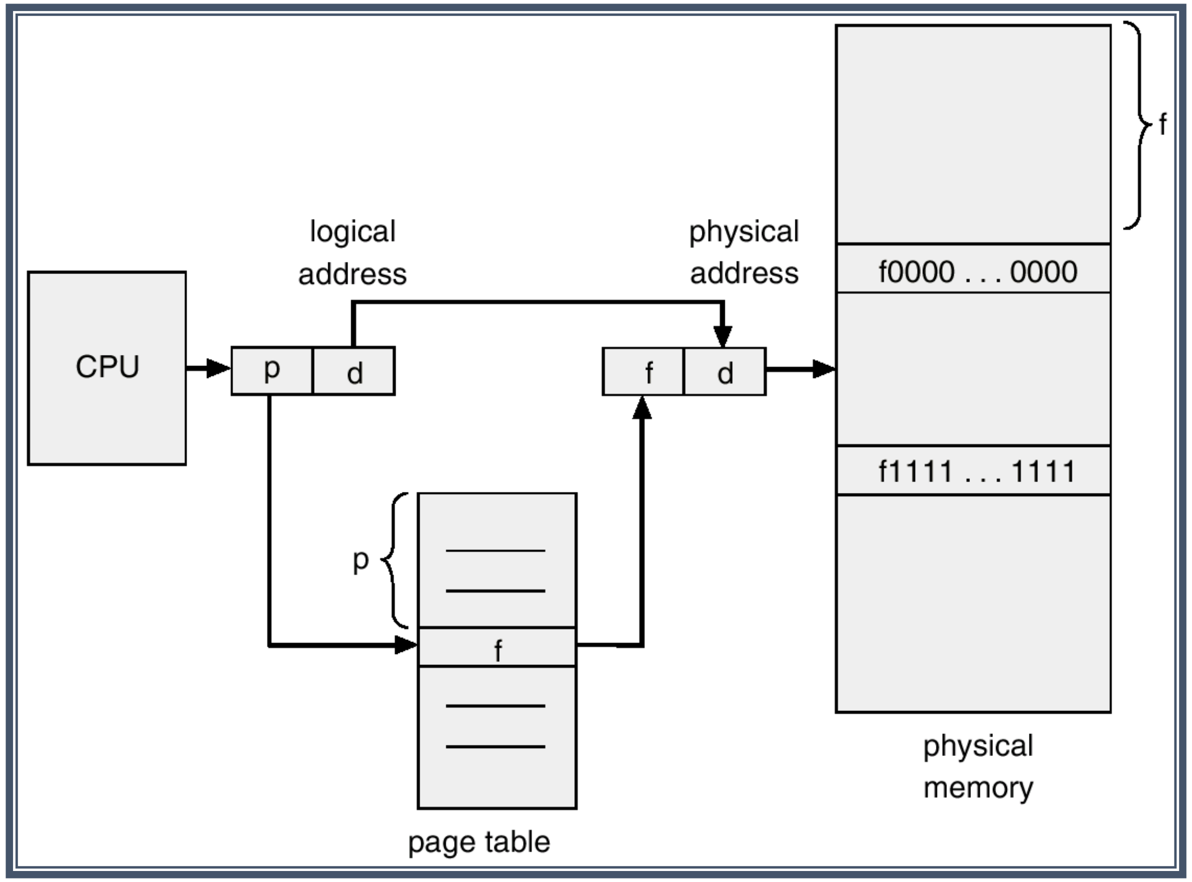
Paging

- Logical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available.
- Divide physical memory into fixed-sized blocks called frames (size is power of 2, between 512 bytes and 8192 bytes).
- Divide logical memory into blocks of same size called pages.
- Keep track of all free frames.
- To run a program of size *n* pages, need to find *n* free frames and load program.
- Set up a page table to translate logical to physical addresses.

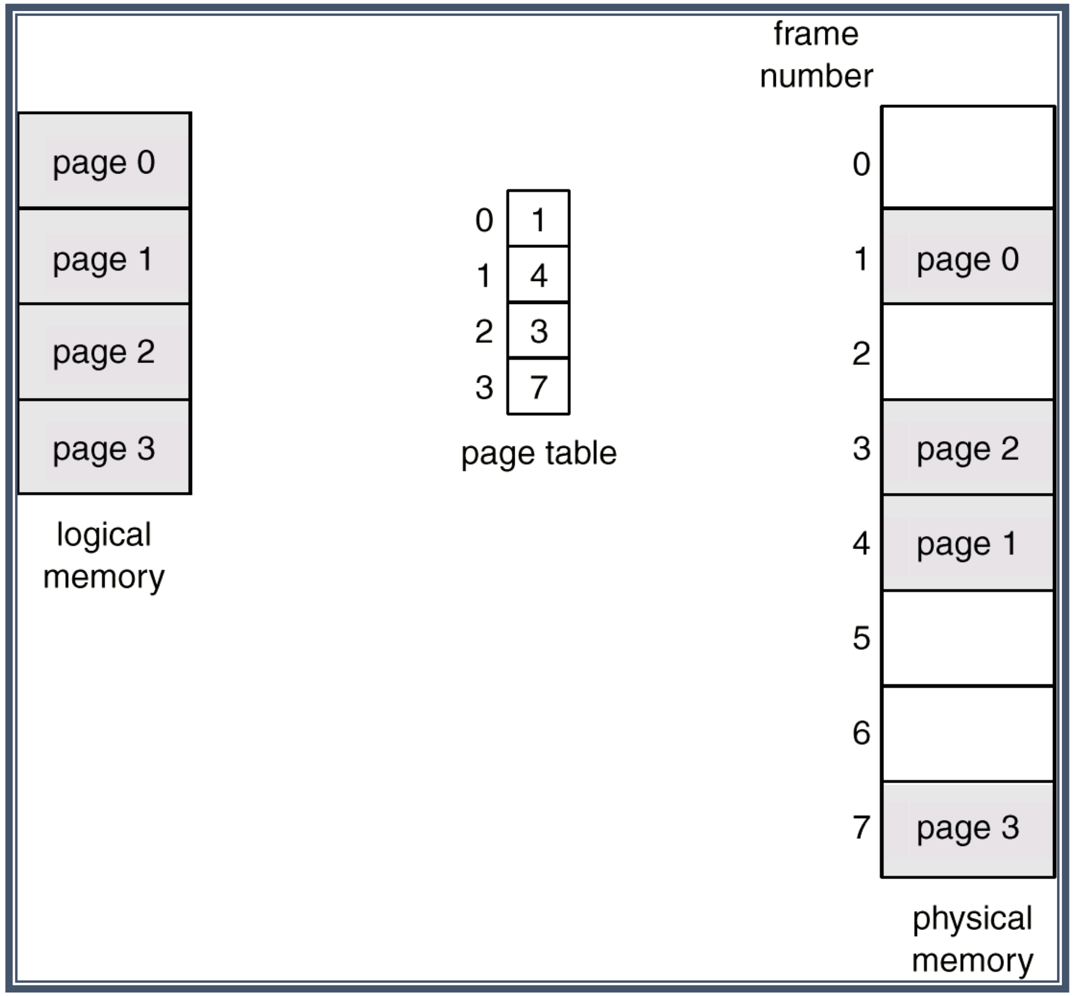
Address Translation Scheme

- Address generated by CPU is divided into:
  - *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.

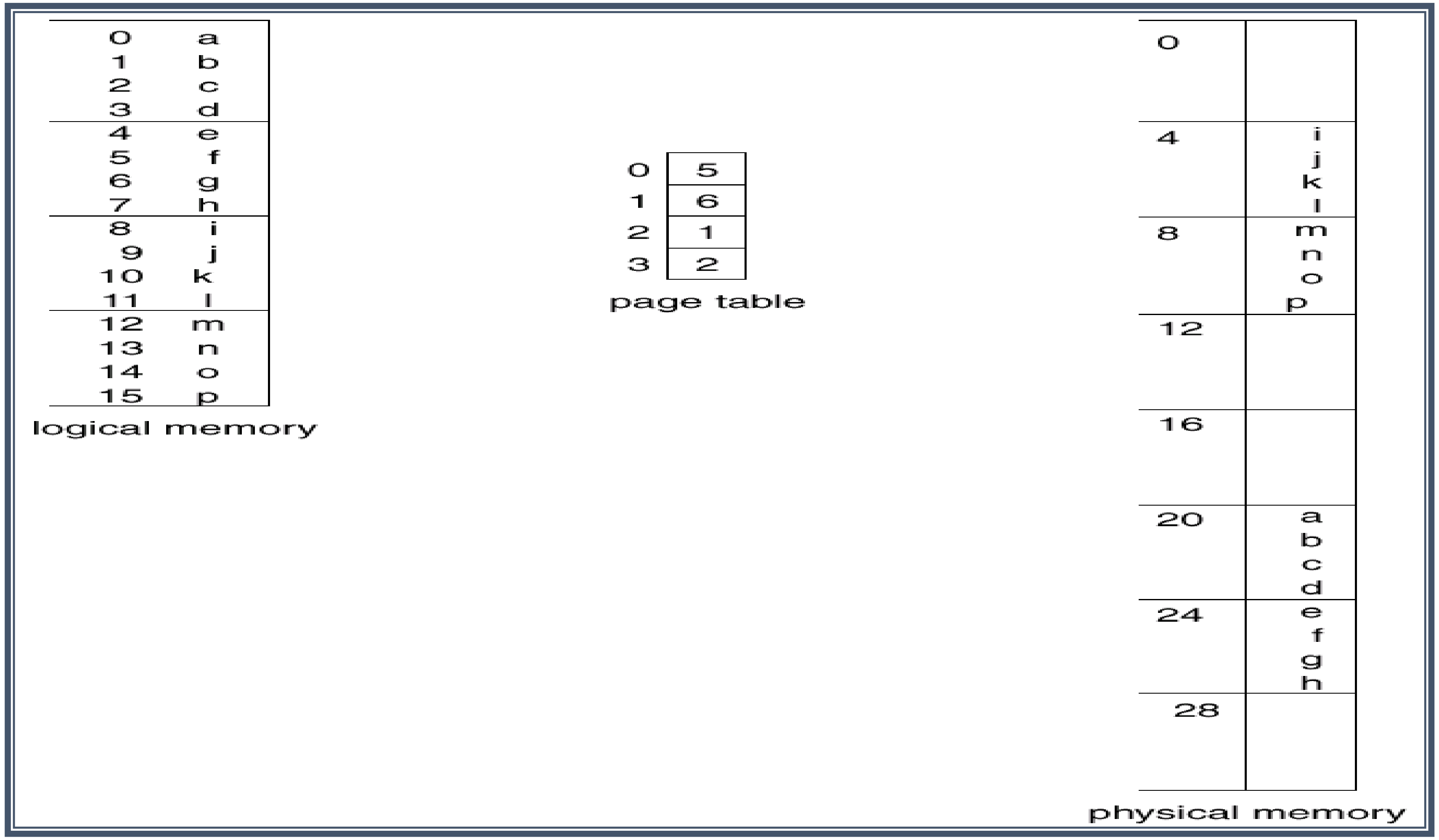
Address Translation Architecture



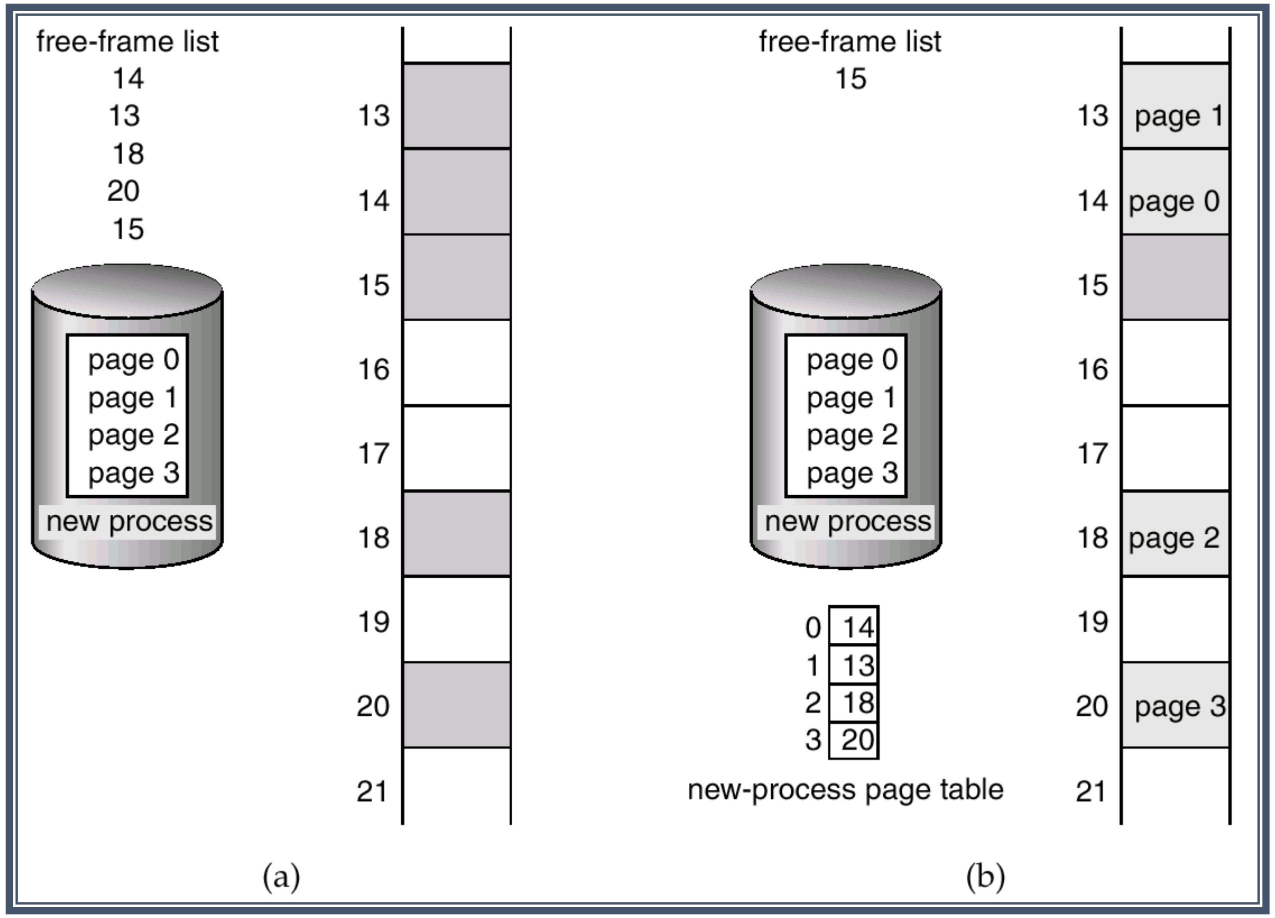
Paging Example



Paging Example



Free Frames



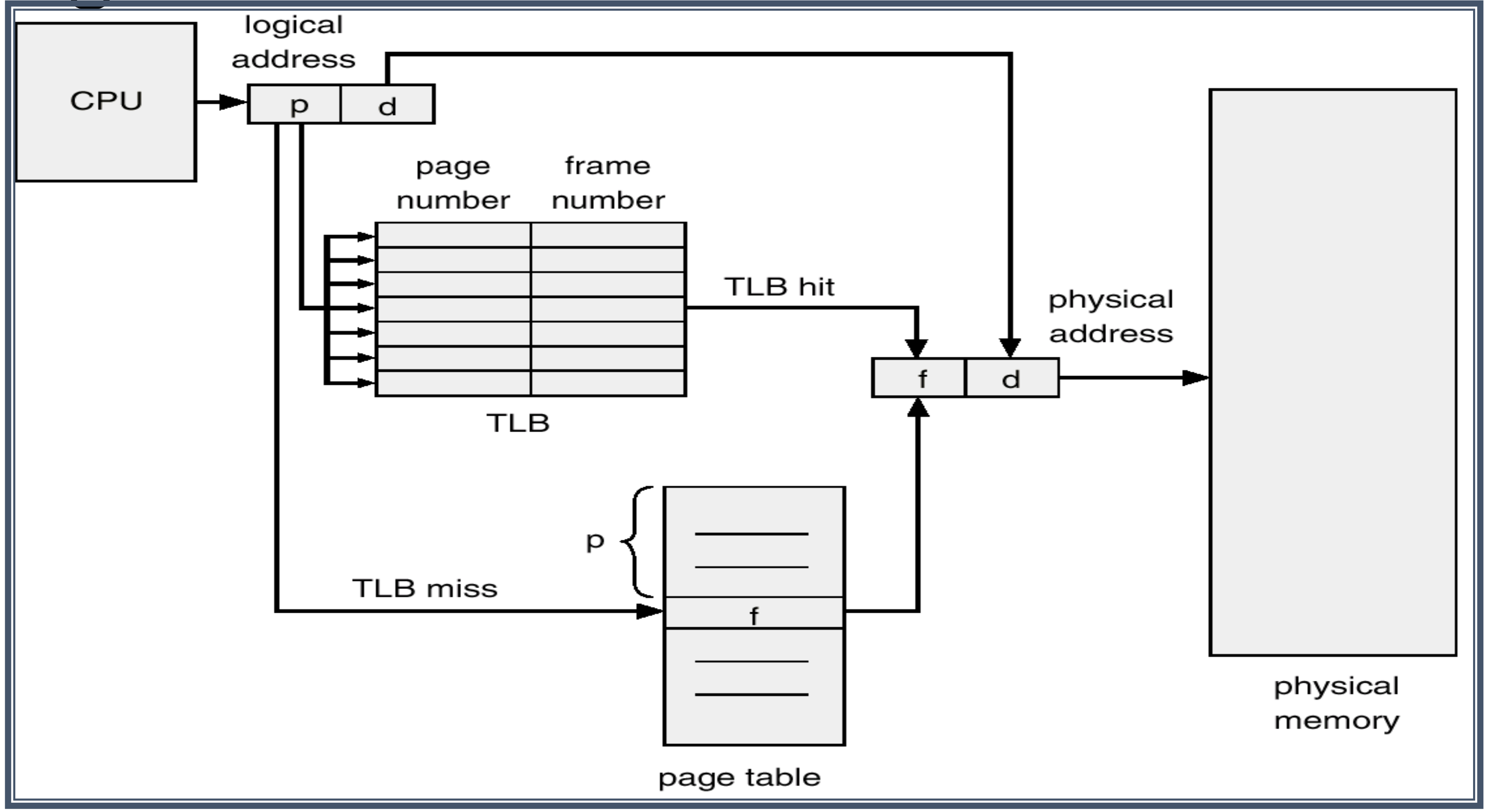
Before allocation

After allocation

Implementation of Page Table

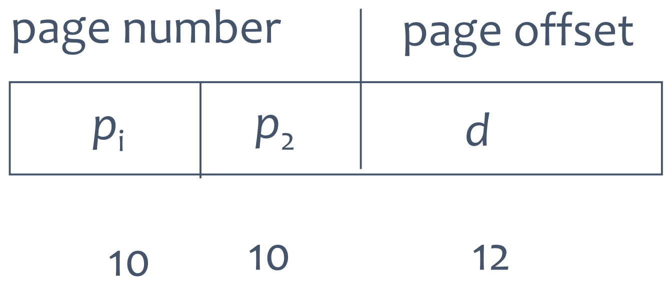
- Page table is kept in main memory.
- *Page-table base register (*PTBR) points to the page table.
- *Page-table length register* (PRLR) indicates size of the page table.
- In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction.
- The two memory access problem can be solved by the use of a special fast-lookup hardware cache called *associative memory* or *translation look-aside buffers (TLBs)*

Paging Hardware With TLB



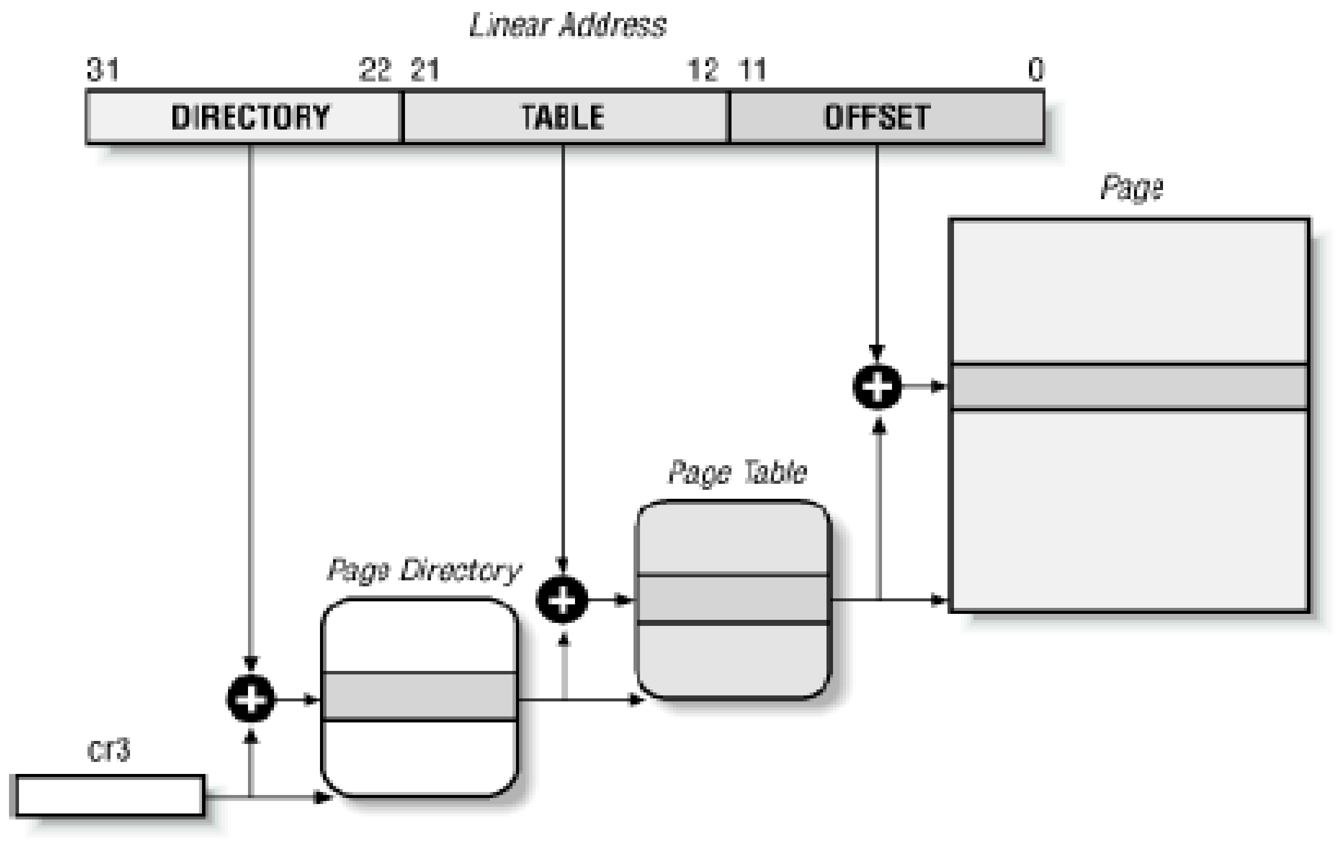
Two-Level Paging Example

- 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:

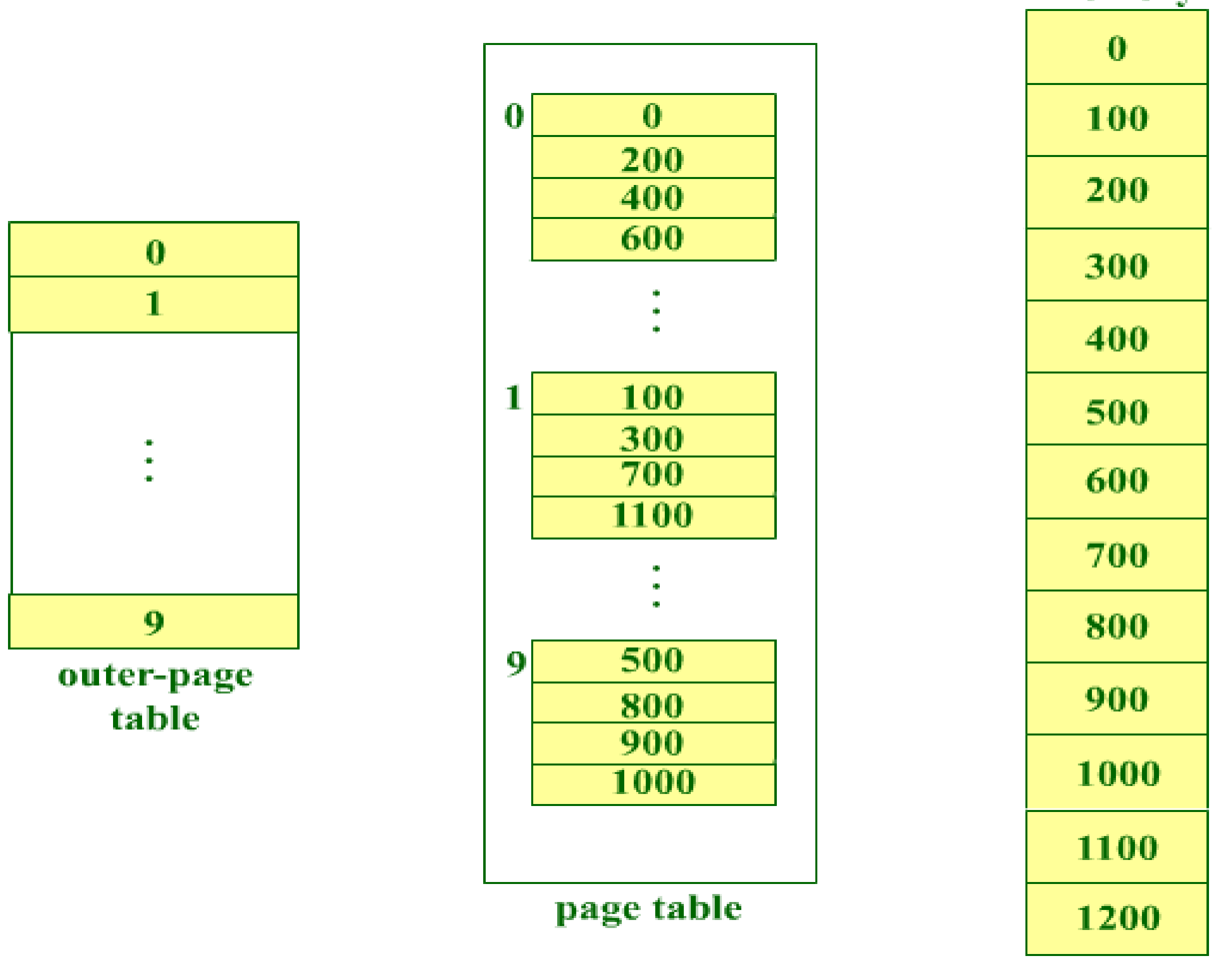


- where pi is an index into the outer page table, and p2 is the displacement within the page of the outer page table.

Two-Level Page-Table Scheme

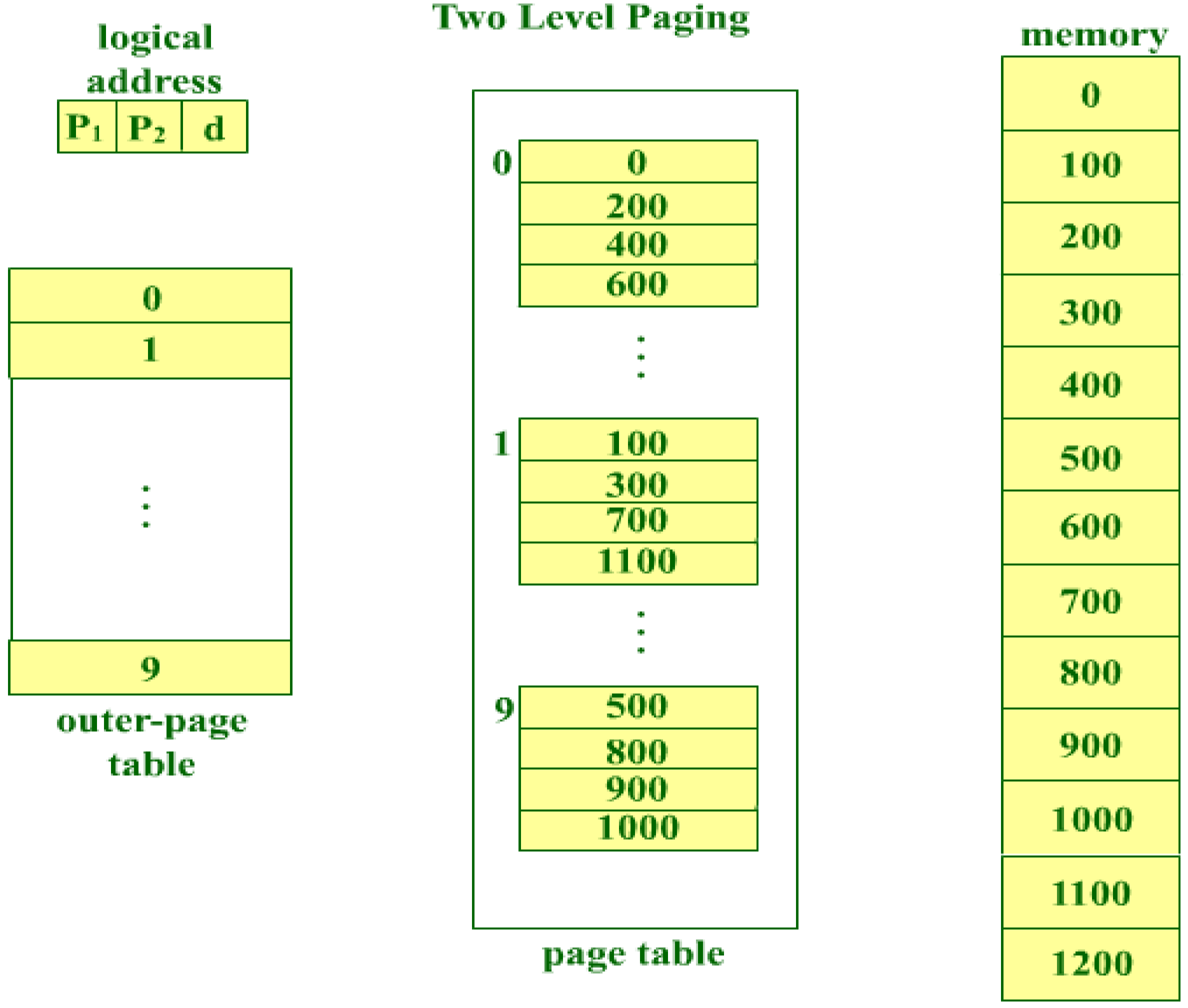


Two Level Paging



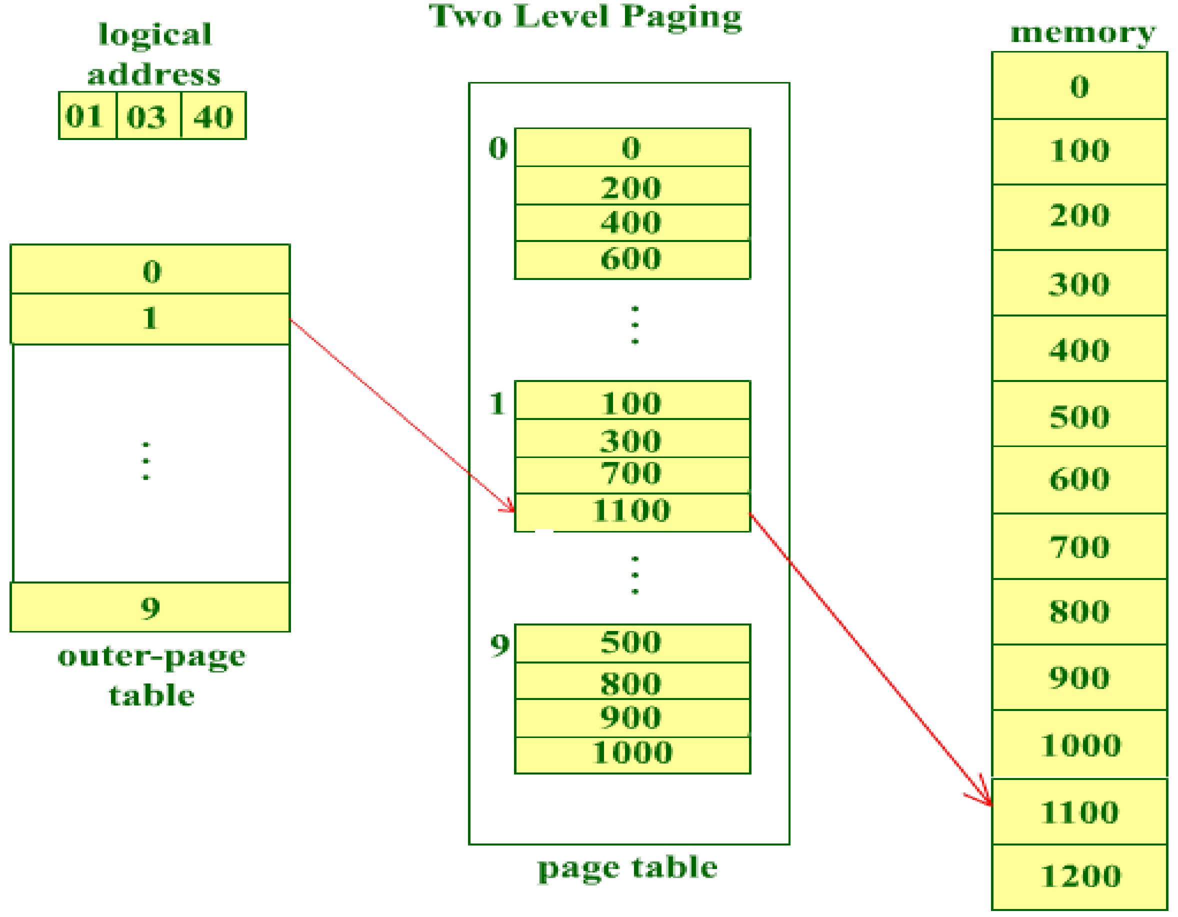
In a two level page table, the page table itself is divided into pages.

Two Level Paging



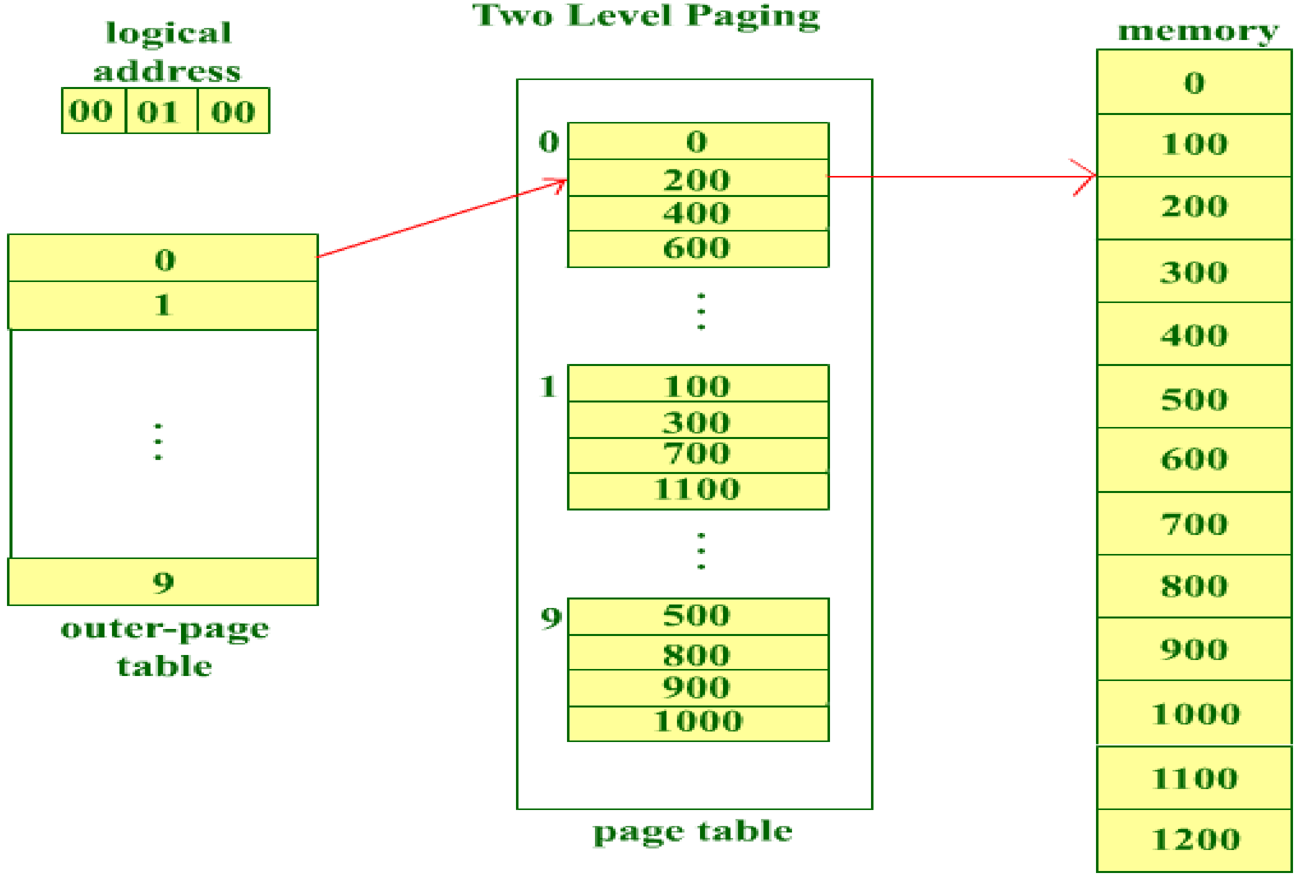
A logical address is divided into an outer page number (P_1), a page displacement within the page of the outer page (P_2), and a page offset (d).

Two Level Paging



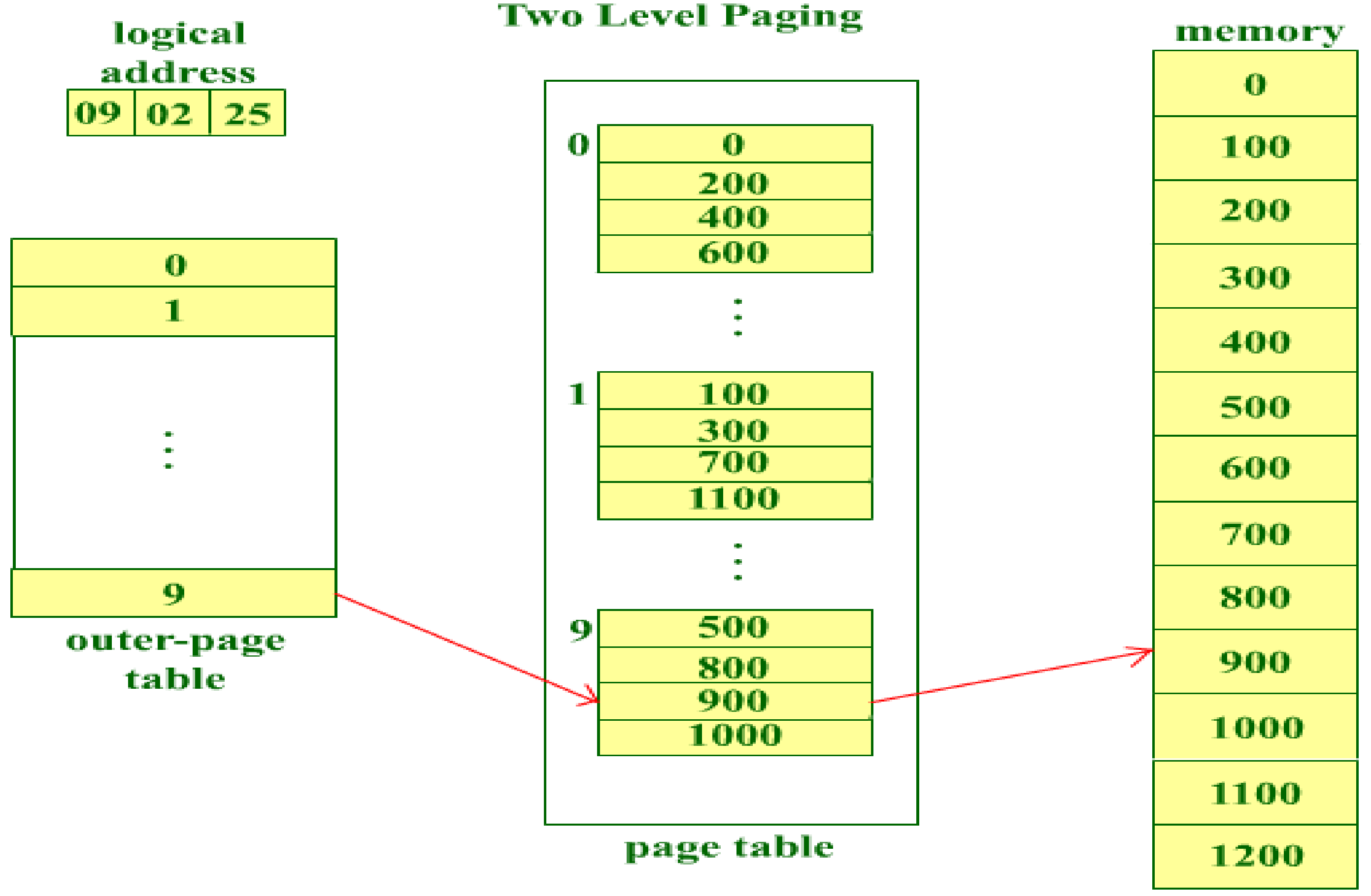
'or example, if the CPU produces logical address 010340, it will reference uter page 01, which points to page 1 in the page table. Then we access offset 3 within the page to obtain frame number 1100. We then use the offset to roduce physical memory address 1140.

Two Level Paging



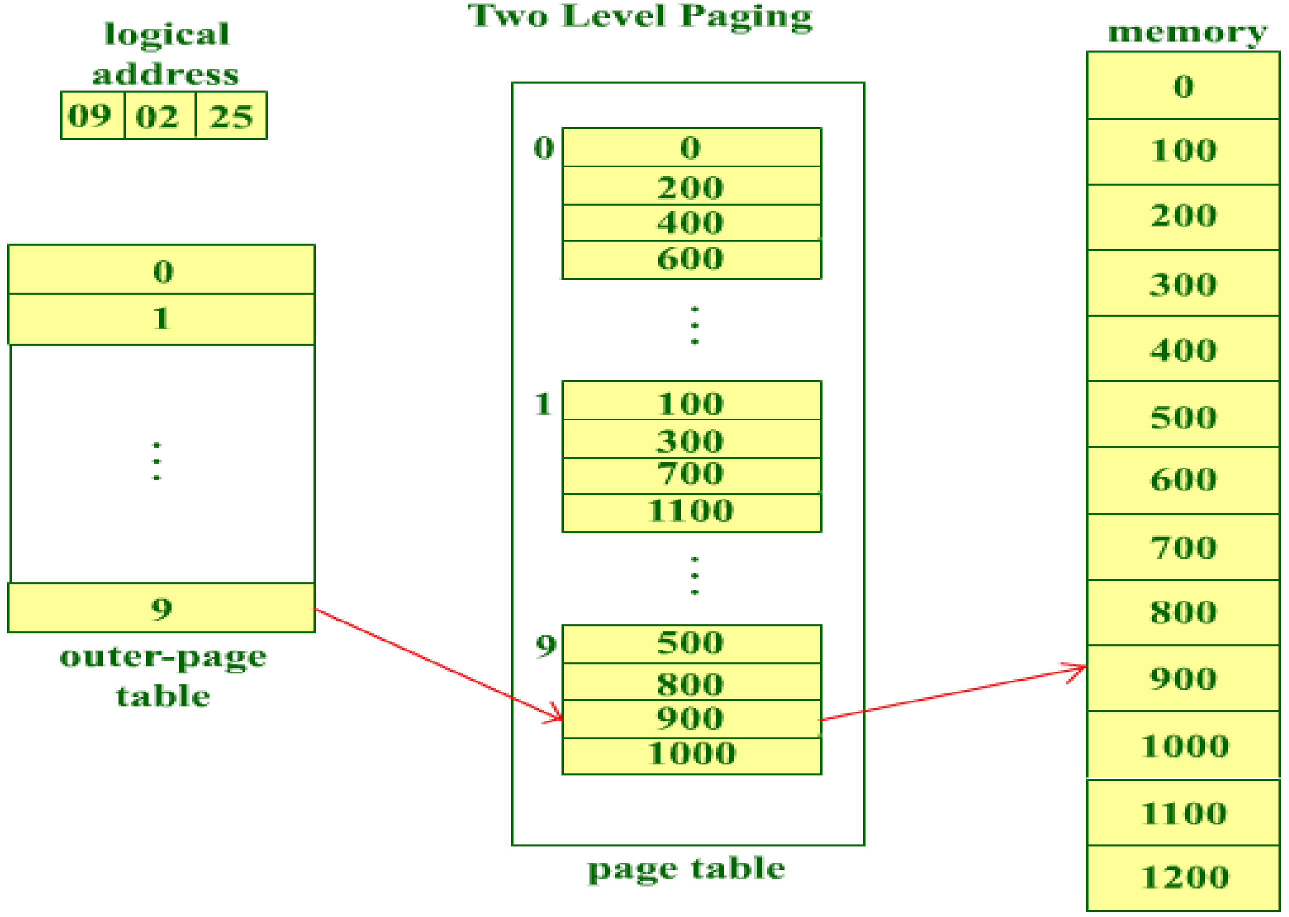
If the CPU produces logical address 000100, it will reference outer page 00, which points to page 0 in the page table. Then we access offset 01 within the page to obtain frame number 200. We then use the offset to produce physical memory address 200.

Two Level Paging



If the CPU produces logical address 090225, it will reference outer page 09, which points to page 9 in the page table. Then we access offset 02 within the page to obtain frame number 900. We then use the offset to produce physical memory address 925.

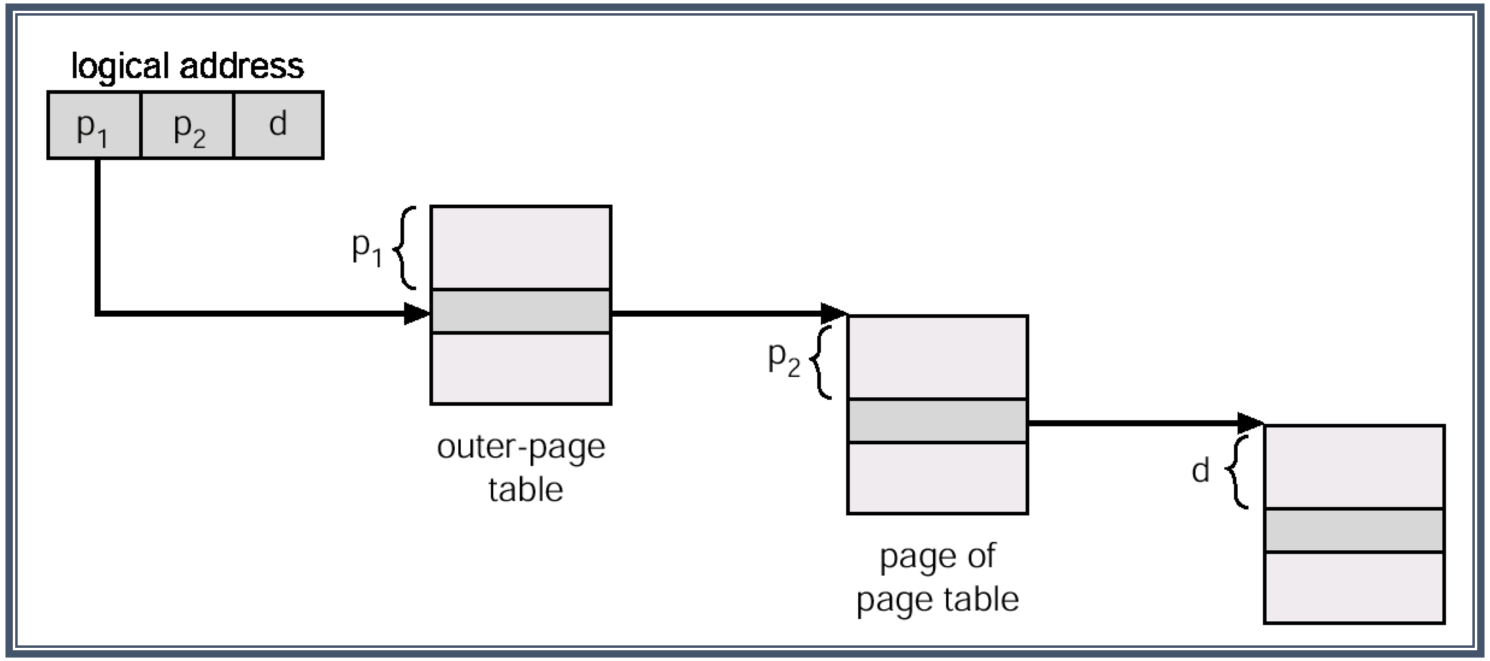
Two Level Paging



CPU produces logical address 090225, it will reference outer page 09,
ı points to page 9 in the page table. Then we access offset 02 within the
to obtain frame number 900. We then use the offset to produce physical

Address-Translation Scheme

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


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