XII

Virtual Memory Technologies
And
Virtual Addressing

Virtual Memory

- Broad concept
- Hides the details of the underlying physical memory
- Provides a view of memory that is more convenient to a programmer
- Can overcome limitations of physical memory and physical addressing

Virtual Memory Terminology

- *Memory Management Unit* (*MMU*)
  - Hardware unit
  - Provides translation between virtual and physical memory schemes
- *Virtual address*
  - Address generated by processor
  - Translated into corresponding physical address by MMU

Virtual Memory Terminology (continued)

- *Virtual address space*
  - Set of all possible virtual addresses
  - Can be larger or smaller than physical memory
- *Virtual memory system*
  - All of the above

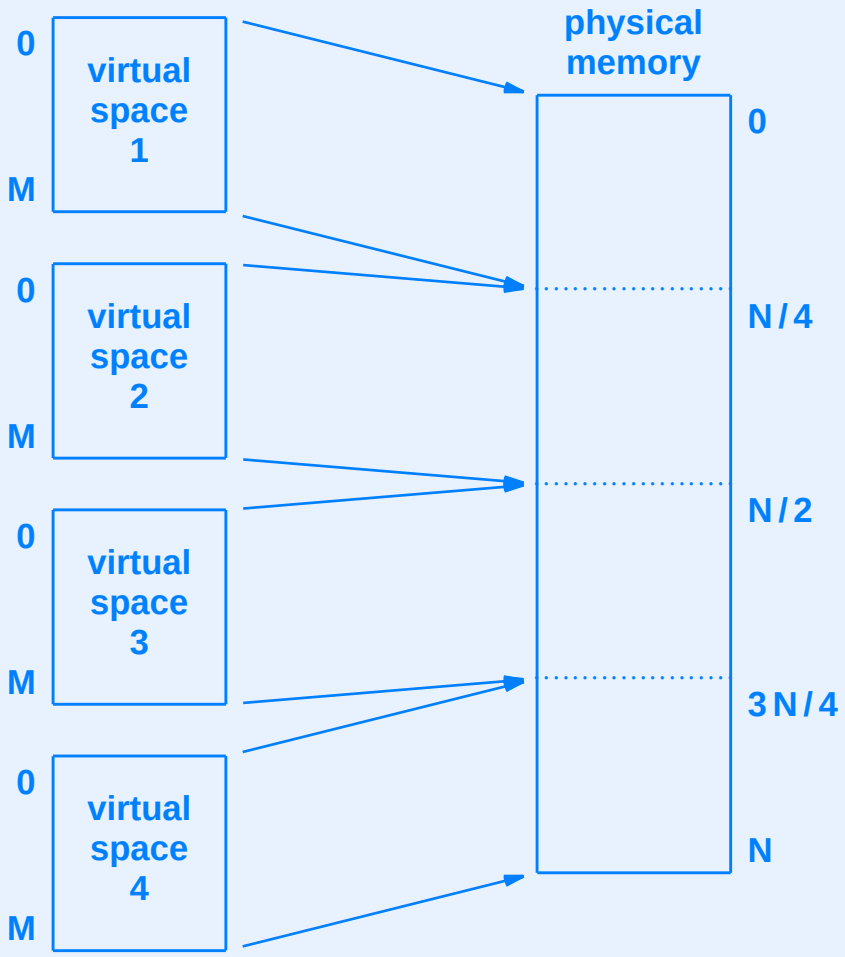
Motivations For Virtual Memory

- Homogeneous integration of hardware
- Programming convenience
- Support for multiprogramming
- Protection of programs and data

Multiple Virtual Spaces And Multiprogramming

- Goal: allow multiple application programs to run concurrently
- Prevent one program from interfering with another
- Trick: provide each program with a separate virtual address space

Illustration Of Four Virtual Address Spaces Mapped To A Single Physical Address Space



Dynamic Address Space Creation

- Processor configures MMU
- Address space mapping can be changed at any time
- Typically
  - Access to MMU restricted to operating system
  - OS runs in *real mode* (access to physical address space)
  - Changes to virtual memory only affect application programs

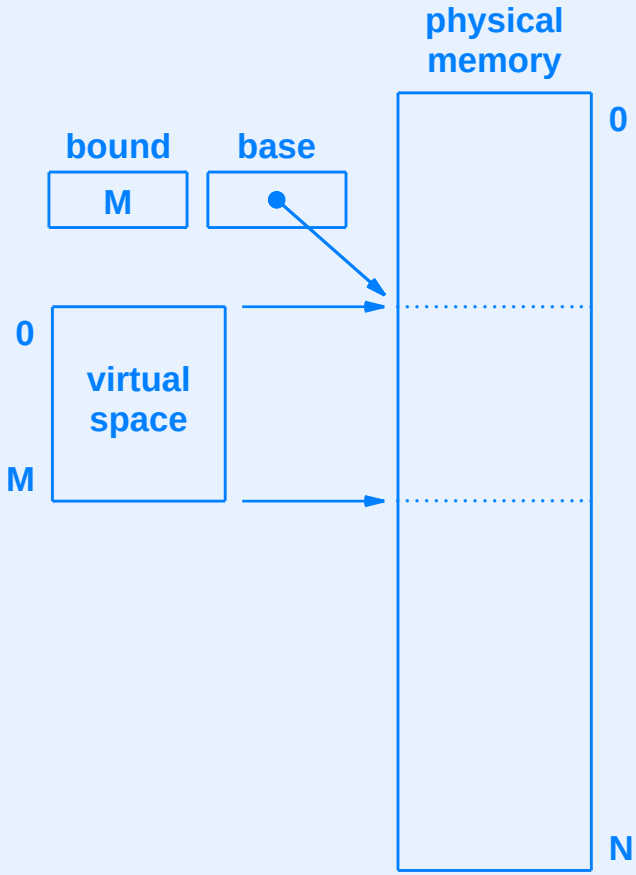
Technologies For Dynamic Address Space Manipulation

- Base-bound registers
- Segmentation
- Demand paging

Base-Bound Registers

- Two hardware registers in MMU
- Base register specifies starting address
- Bound register specifies size of address space
- Values changed by operating system
  - Set before application runs
  - Changed by operating system when switching to another application

Illustration Of Virtual Memory Using Base-Bound Registers



Protection

- Multiple applications each allocated separate area of physical memory
- OS sets base-bound registers before application runs
- MMU hardware checks each memory reference
- Reference to any address outside the valid range results in an error

The Concept Of Protection

A virtual memory system that supports multiprogramming must also provide protection that prevents one program from reading or altering memory that has been allocated to another program.

Demand Paging

- Alternative to segmentation and base-bounds
- Most popular virtual memory technology
- Divides program into fixed-size pieces called *pages*
- No attempt to align page boundary with procedure
- Typical page size 4K bytes

Support Needed For Demand Paging

- Hardware that handles address mapping and detects missing pages
- Software that moves pages between external store and physical memory

Paging Hardware

- Part of MMU
- Intercepts each memory reference
- If referenced page is present in memory, translate address
- If referenced page not present in memory, generate a *page fault* (error condition)
- Allow operating system to handle the fault

Demand Paging Software

- Part of the operating system
- Works closely with hardware
- Responsible for overall memory management
- Determines which pages of each application to keep in memory and which to keep on disk
- Records location of all pages
- Fetches pages on demand
- Configures the MMU

Page Replacement

- Initially
  - Applications reference pages
  - Each referenced page is placed in physical memory
- Eventually
  - Memory is full
  - Existing page must be written to disk before memory can be used for new page
- Choosing a page to expel is known as *page replacement*
- Should replace a page that will not be needed soon

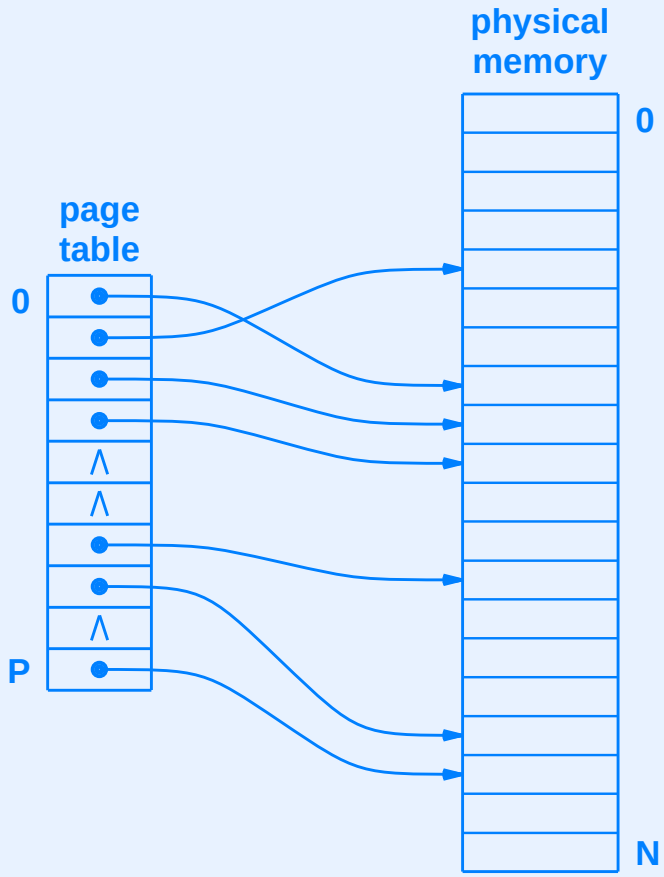
Paging Terminology

- *Page*: fixed-size piece of program's address space
- *Frame*: slot in memory exactly the size of one page
- *Resident*: a page that is currently in memory
- *Resident set*: pages from a given application that are present in memory

Paging Data Structure

- Page table
  - One per application
  - Think of each as one-dimensional array indexed by page number
  - Stores the location of each page in the application (either in memory or on disk)

Illustration Of A Page Table

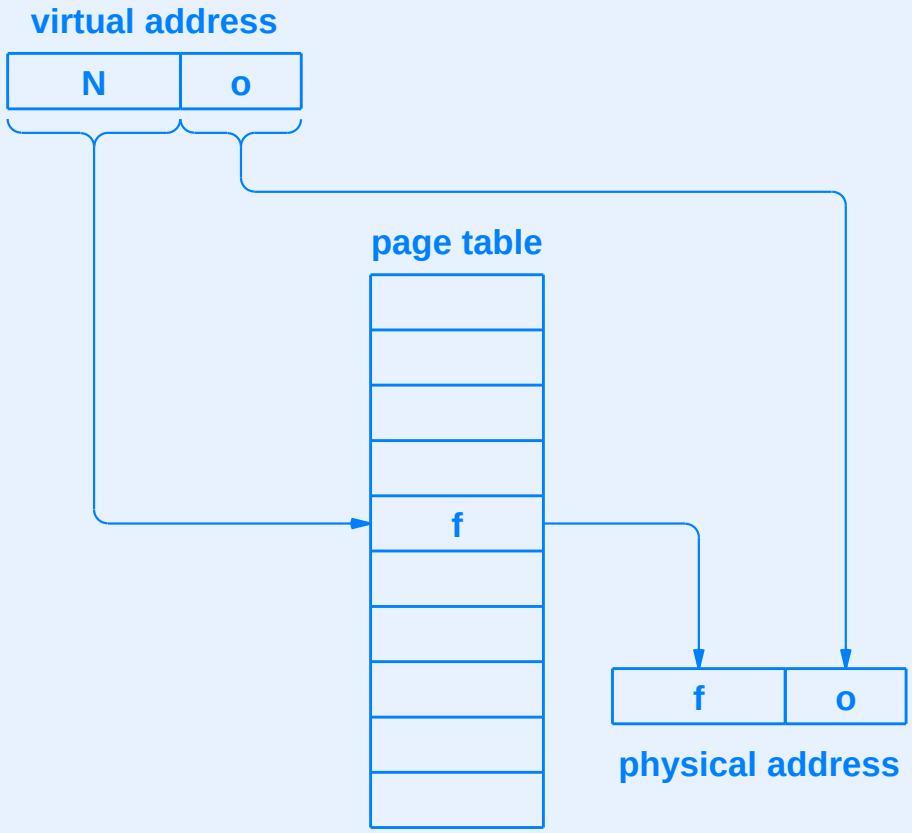


- Typical system has 4K bytes per page

Address Translation

- Given virtual address *V*, find physical memory address *P*
- Three conceptual steps
  - Determine the number of the page on which address *V* lies
  - Use the page number as an index into the page table to find the location in memory that corresponds to the first byte of the page
  - Determine how far into the page *V* lies, and convert to a position in the frame in memory

Illustration Of Translation With MMU Hardware



Presence, Use, And Modified Bits

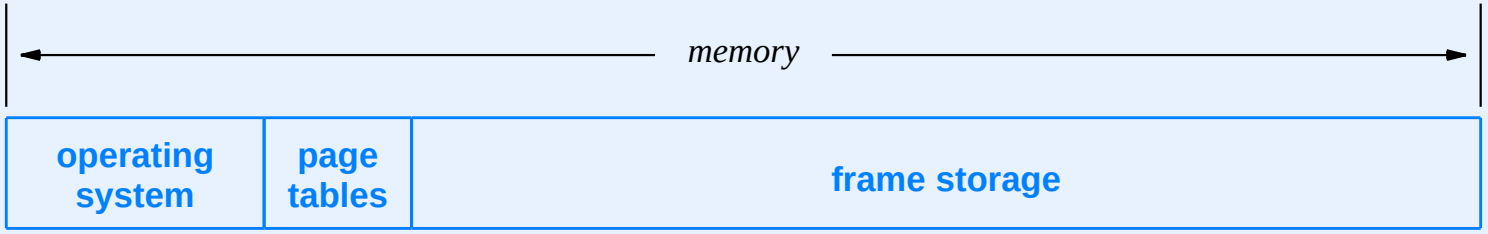
- Found in most paging hardware
- Shared by hardware and software
- Purpose of each bit:

Control Bit	Meaning
Presence bit	Tested by hardware to determine whether page is currently present in memory
Use bit	Set by hardware whenever page is referenced
Modified bit	Set by hardware whenever page is changed

Page Table Storage

- Page tables occupy space
- Two possibilities for page table storage
  - In MMU
  - In main memory

Illustration Of Page Tables Stored In Physical Memory



Paging Efficiency

- Paging must be used
  - For each instruction fetch
  - For each data reference
- Can become a bottleneck
- Must be optimized

Translation Lookaside Buffer (TLB)

- Hardware mechanism
- Optimizes paging system
- Form of Content Addressable Memory (CAM)
- Stores pairs of

  (virtual address, physical address)

- If mapping in TLB
  - No page table reference needed
  - MMU can return mapping quickly

In Practice

- Virtual memory system without TLB is unacceptable
- TLB works well because application programs tend to reference given page many times

The Importance Of A TLB

A special high-speed hardware device called a Translation Lookaside Buffer (TLB) is used to optimize performance of a paging system. A virtual memory that does not have a TLB can be unacceptably slow.

Summary

- Virtual memory systems present illusion to processor and programs
- Many virtual memory architectures are possible
- Examples include
  - Hiding details of word addressing
  - Create uniform address space that spans multiple memories
  - Incorporate heterogeneous memory technologies into single address space

Summary
(continued)

- Virtual memory offers
  - Convenience for programmer
  - Support for multiprogramming
  - Protection
- Three technologies used for virtual memory
  - Base-bound registers
  - Segmentation
  - Demand paging

Summary
(continued)

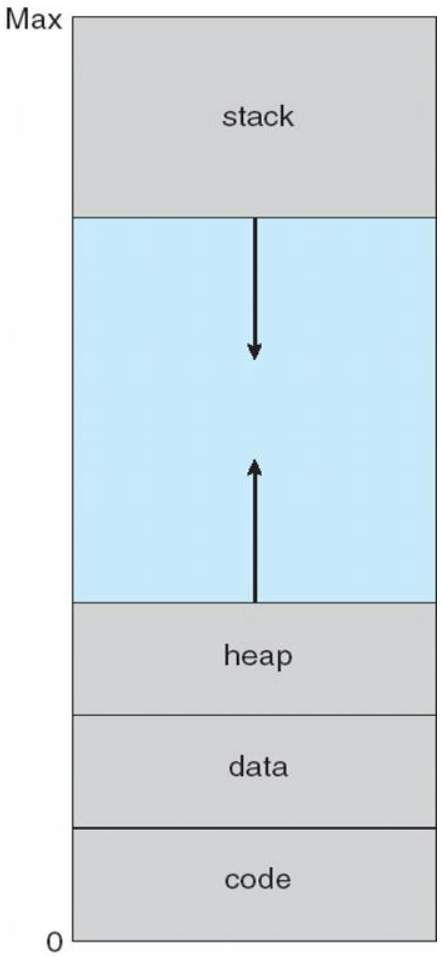
- Demand paging
  - The most popular technology
  - Combination of hardware and software
  - Uses page tables to map virtual addresses to physical addresses
  - High-speed lookup mechanism known as TLB makes demand paging practical

Virtual vs. Physical Address Space

- Each process has its own virtual address space, which may be larger than the physical address space (namely size of RAM).

- The concept of a virtual address space that is bound to a separate **physical address space** is central to proper memory management
  - **Virtual address** – generated by the CPU
  - **Physical address** – address seen by the memory controller

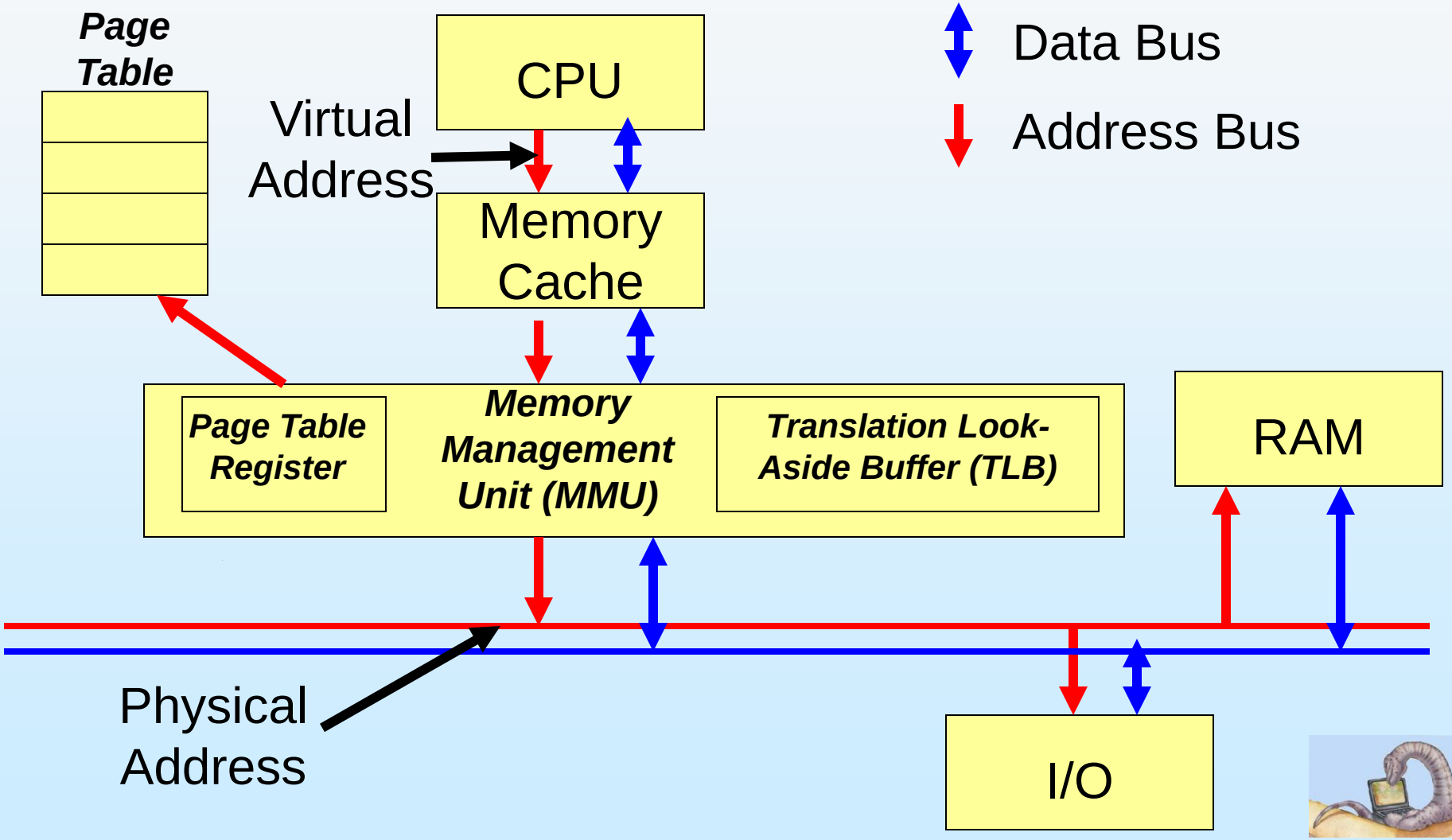
Virtual Address Space of a Process



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 *virtual* addresses; it never sees the real physical addresses
- CPU generates virtual addresses

The Memory Management Unit



Paging

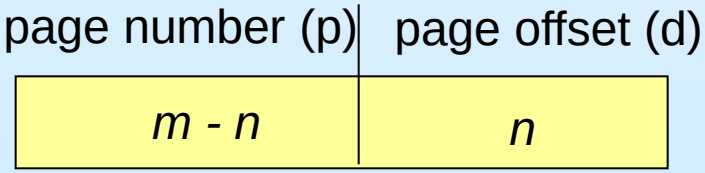
- Physical address space of a process can be noncontiguous; process is allocated physical memory whenever the latter is available
- Divide physical memory into fixed-sized blocks called **frames** (size is power of 2, between 512 bytes and 8,192 bytes)
- Divide virtual memory into blocks of same size 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 virtual to physical addresses

Paging

- The Virtual Memory system will keep in memory the pages that are currently in use.
- It will leave in disk the memory that is not in use.

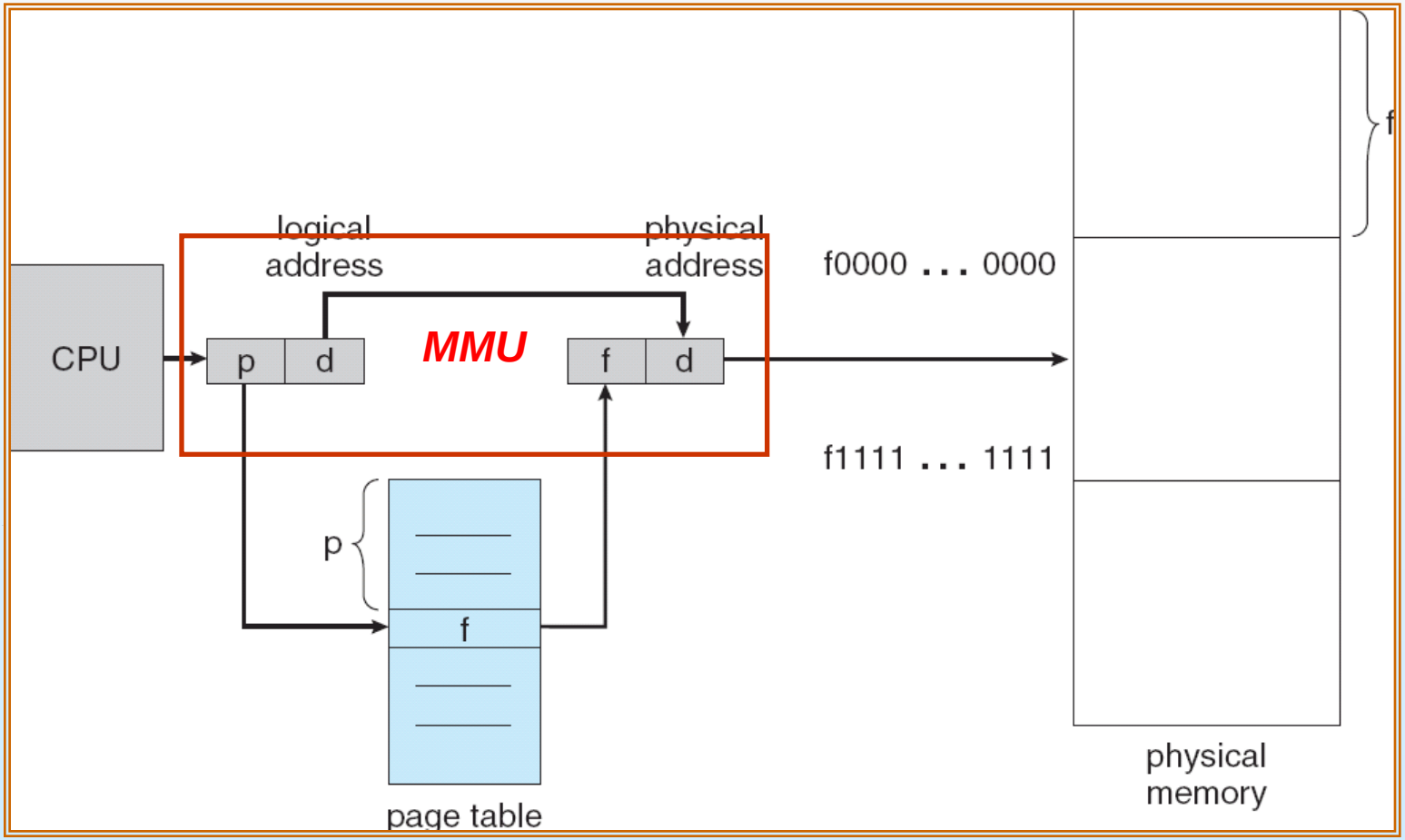
Address Translation Scheme

- Virtual 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

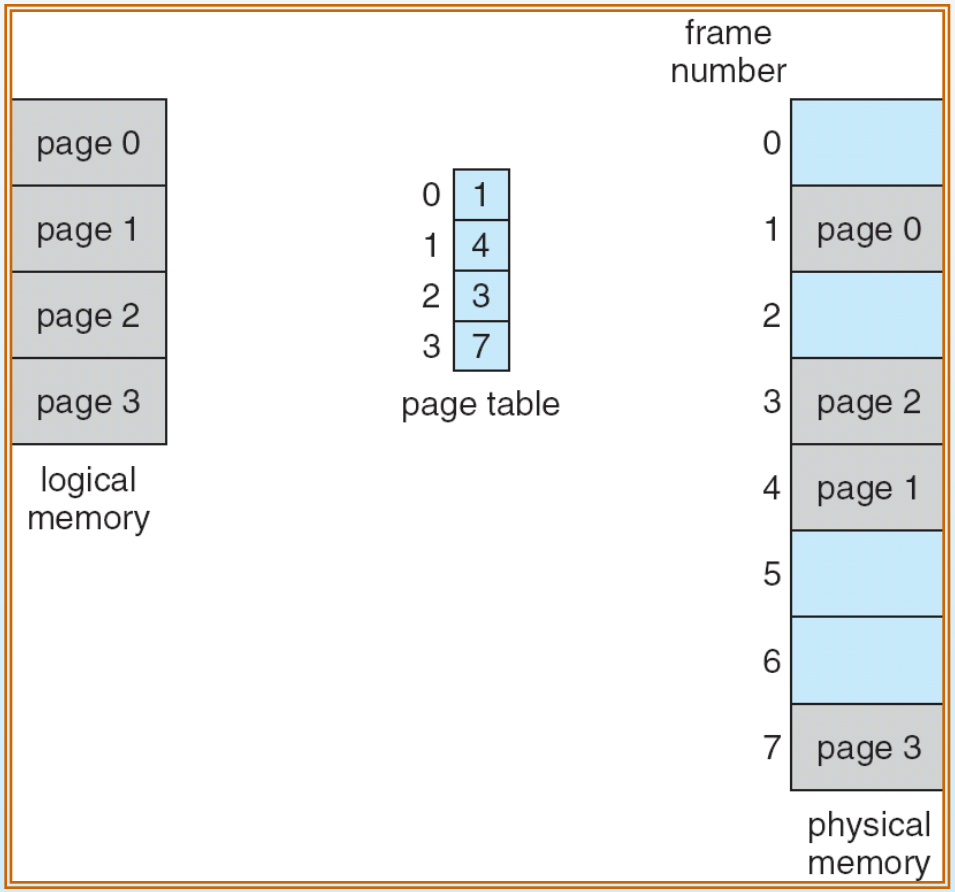


For given virtual address space 2^m *and page size* 2^n

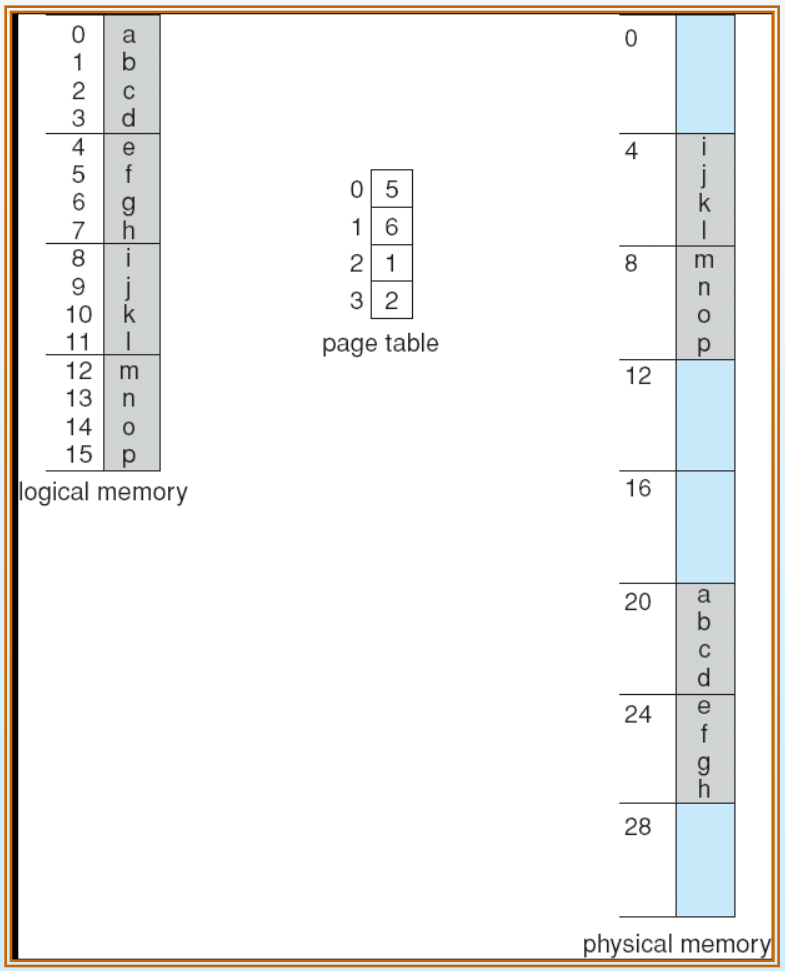
Paging Hardware



Paging Model of Virtual (i.e. Logical) and Physical Memory

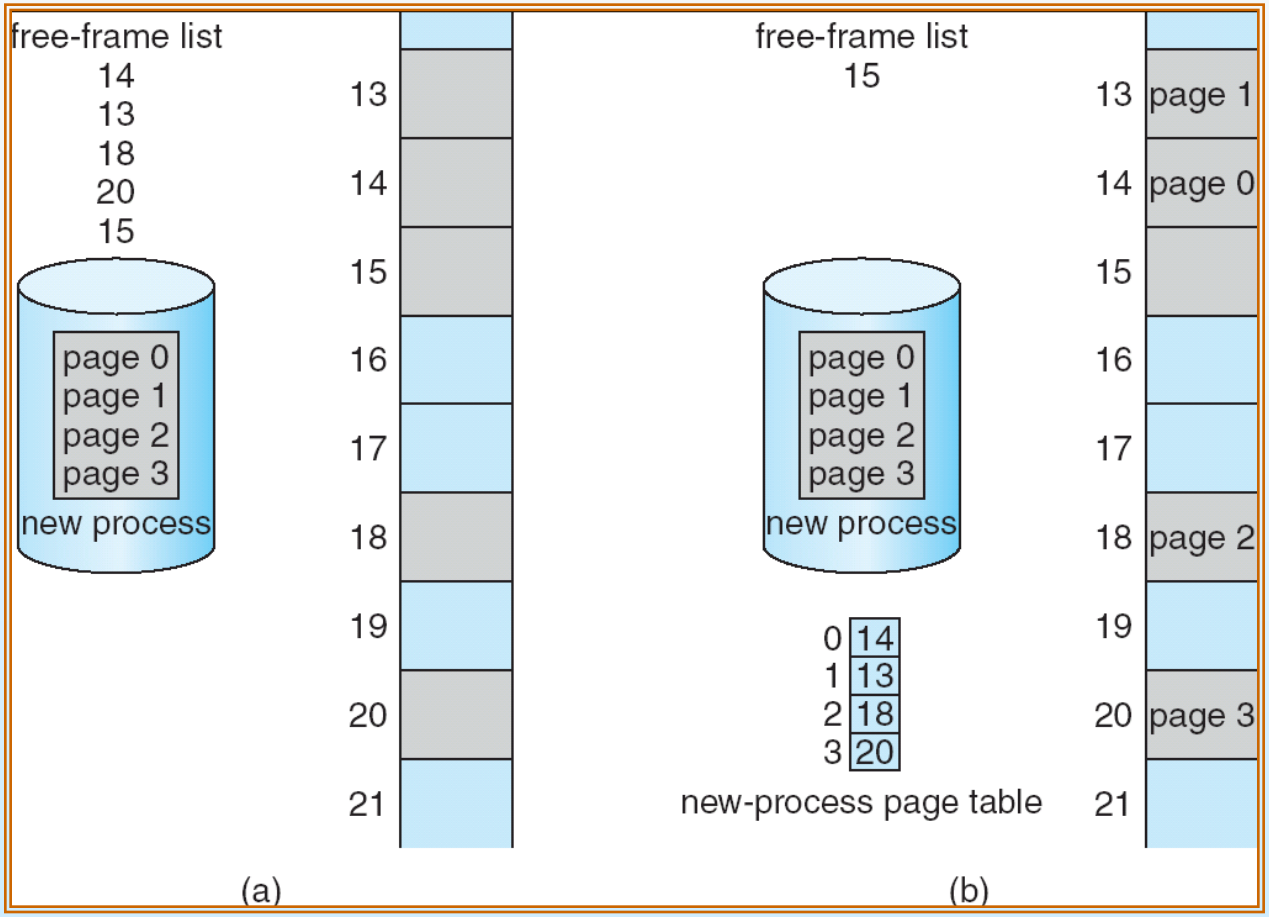


Paging Example



32-byte memory and 4-byte pages

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)**

TLB via Associative Memory

- Associative memory – parallel search

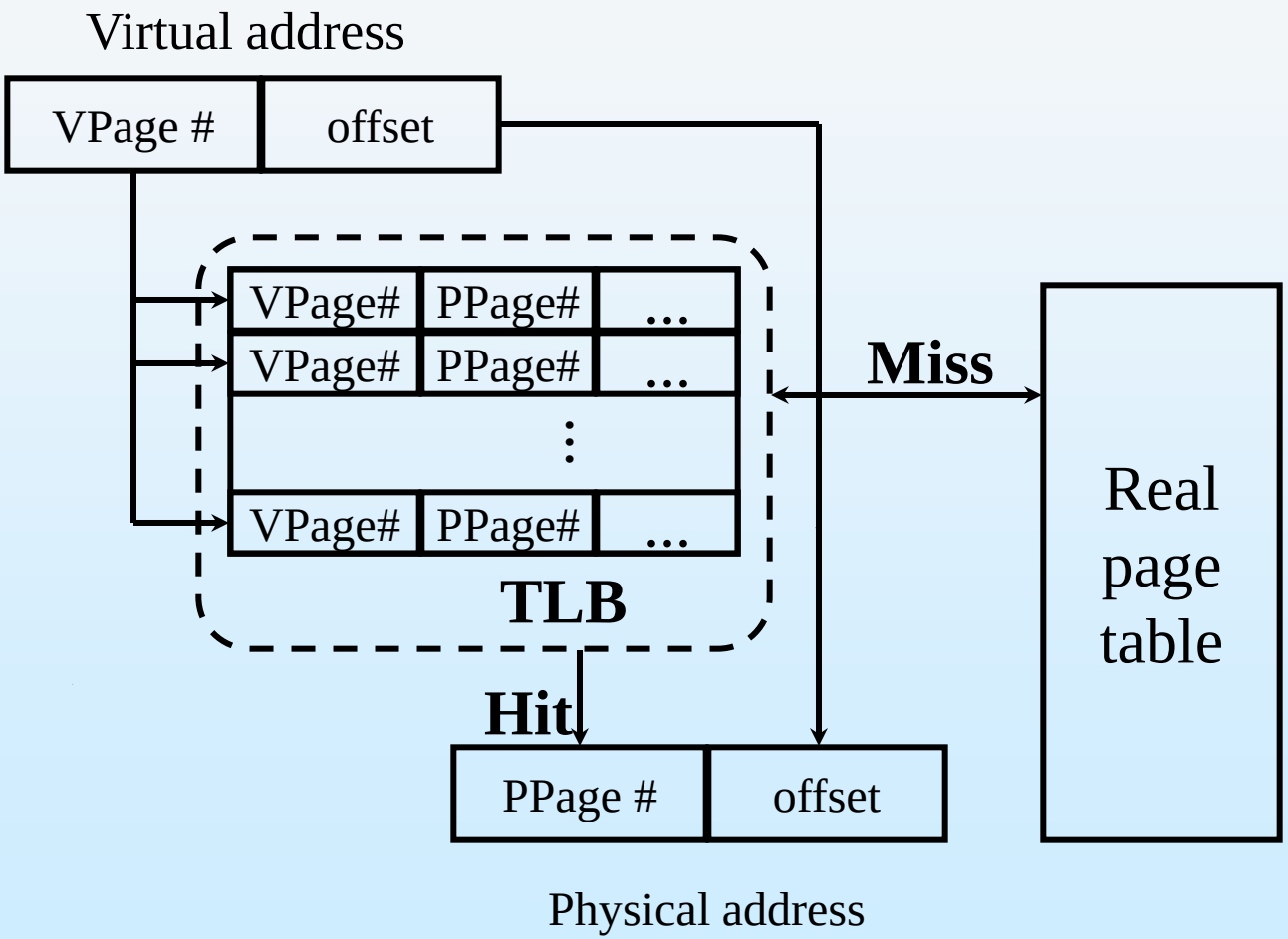
Page #	Frame #

What are the differences between the page table and the TLB?

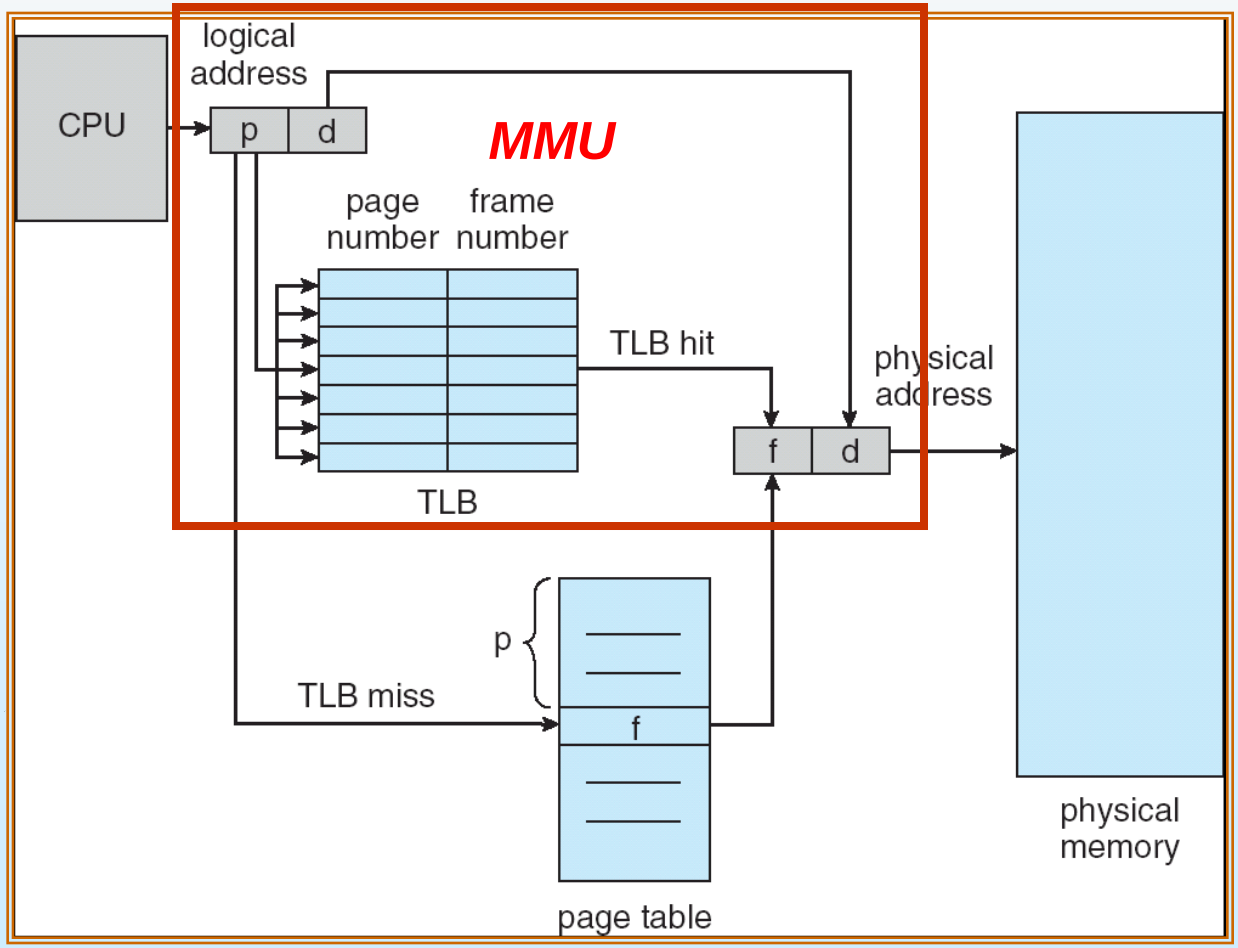
Address translation (p, d)

- If p matches a page # in TLB, get frame # from the same entry in TLB
- Otherwise get frame # from page table in memory

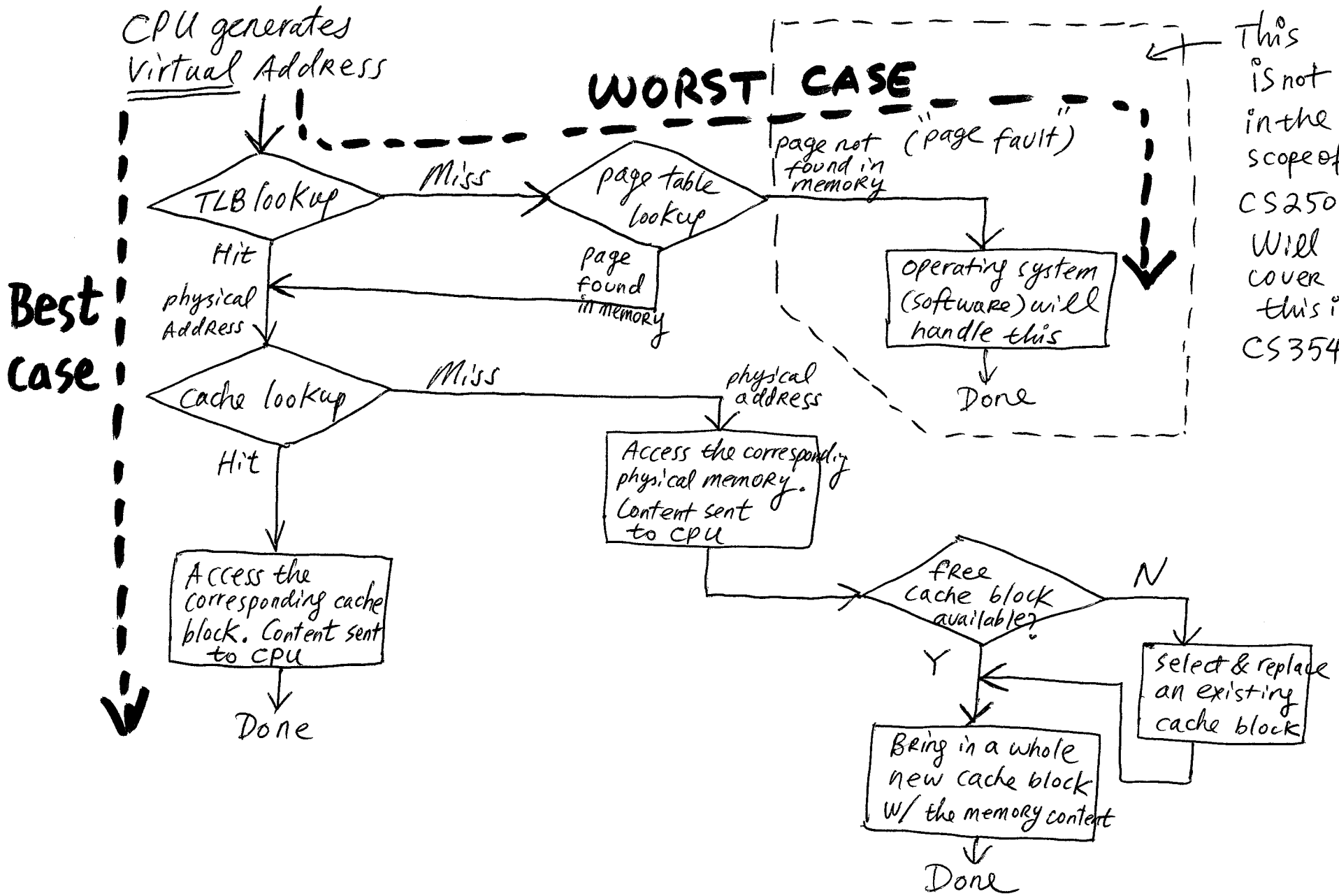
Translation Look-aside Buffer (TLB)



Paging Hardware With TLB



Virtual Memory: Putting things (cache, memory, TLB, page table ...) Together

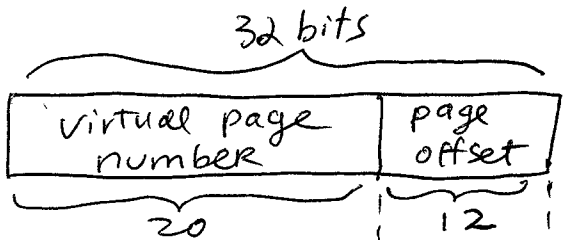


An Example:

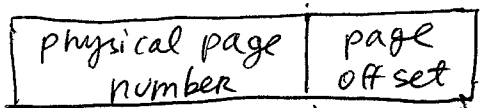
- 32-bit architecture
- page size = 4KB (2^{12})
- RAM (physical memory) = 256MB (2^{28} bytes)
- Cache size = 1MB (2^{20})
  - Two-way set associative
  - Block size = 16 bytes

$\Rightarrow$ Total # of blocks $= 2^{16} \Rightarrow$ Total # of sets $= 2^{15}$

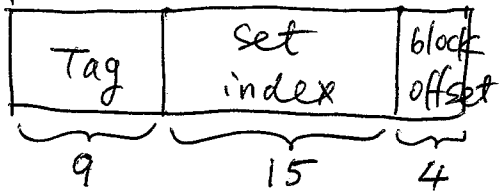
Virtual Address partitioning



physical address partitioning



Same physical address used for cache lookup



TLB: (showing part of it)

Valid	virtual page #	physical page #
1	0x 00001	0x AAAA
1	0x 12345	0x EEEF
1	0xABCDE	0x 0001
⋮		

Given the TLB on the left bottom and virtual address 0x ABCDE 123

Address Translation

Translated into physical address