Virtual Memory

Peter J. Denning

Definition

- Virtual memory is an OS subsystem that automatically manages address spaces and the movement of their pages up and down the memory hierarchy.
- “Virtual”, a term borrowed from optics (“virtual image”)*, refers to the simulation of a RAM large enough to hold the entire address space.

*virtual: you can see it but it is not there
transparent: you cannot see it, but it is there

History

- VM invented in 1959 at University of Manchester to solve the “overlay problem”.
  - On early machines programmers had to program their own up and down operations.
  - Ups and downs were called overlays. Manual overlays were time consuming and error prone.
  - Automation would improve programmer productivity by 2x or 3x or more.

- Their innovations:
  - Distinction between address and location
  - Address space: the virtual memory to be simulated with a smaller RAM, looks contiguous to the CPU
  - Pages and frames
  - Mapping of pages to frames
  - Page fault and associated interrupt
  - Replacement algorithms

- Widespread initial enthusiasm
- Performance issues quickly cast a dark cloud
- Classic 1966 study by Les Belady at IBM clarified many performance issues of page replacement
  - Use bits
  - Early evidence of locality
- In 1966 Denning at MIT invented working set model
  - Analytic theory
  - Exploited locality to be near optimal
  - Prevented thrashing
  - Approximations adopted into all OSs

Addressing

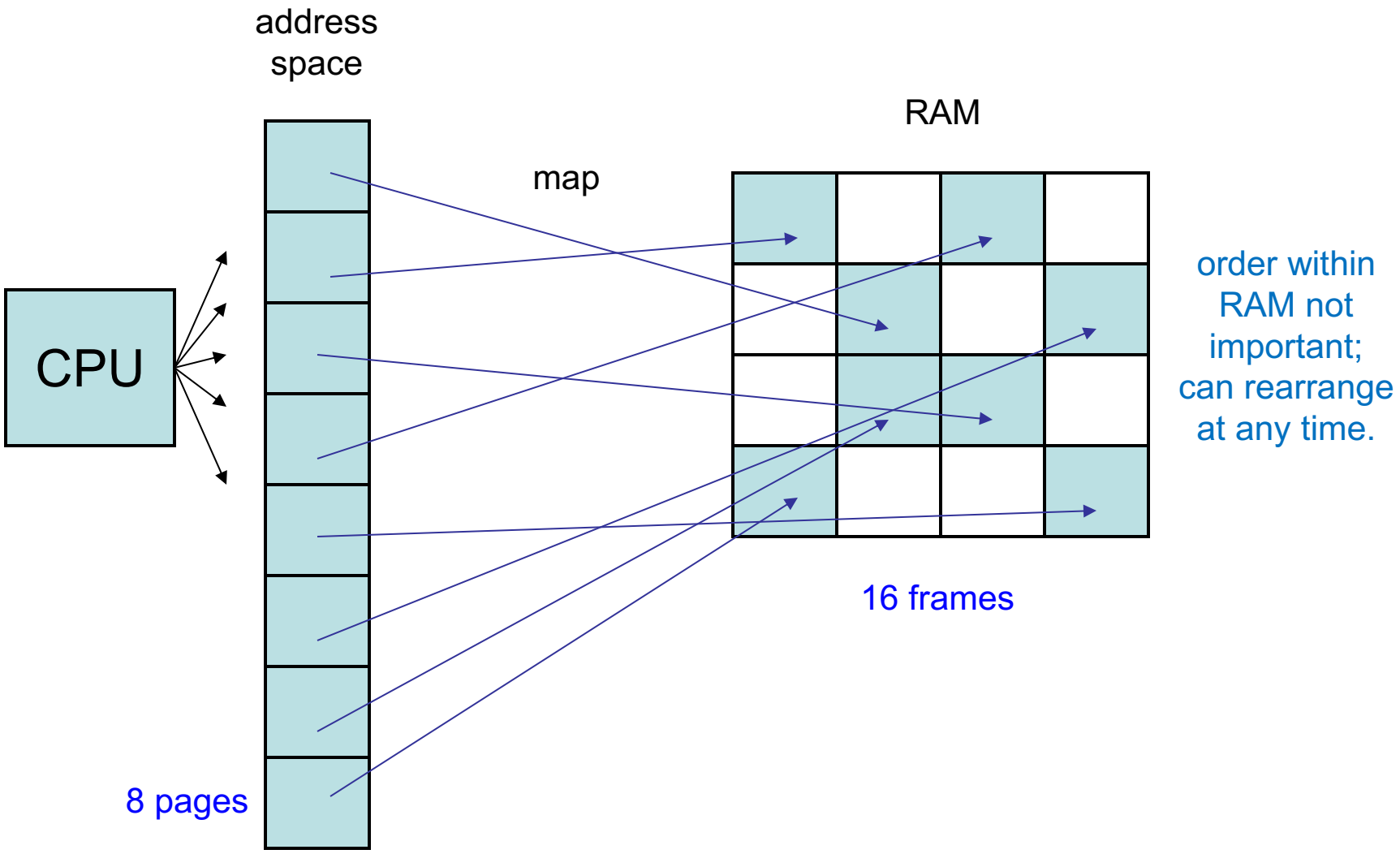
- Basic data elements stored in memory are bytes (8-bits).
- Each byte is stored in a physical memory location.
  - RAM: registers
  - Disk, CD: patches of magnetic or optical surface
- Addresses: bit-strings that identify bytes.
  - Virtual address: generated by the CPU to designate a specific data byte in a contiguous sequence of bytes called Address Space. Example: A CPU with 16-bit addresses can access any item in the contiguous range 0, 1, 2, …, 2^{16}-1.
  - Physical address: designates specific location of a byte in RAM, also numbered in a contiguous sequence. Example: A RAM with 32-bit addresses can access any location in the contiguous range 0, 1, 2, …, 2^{32}-1.

- Address map: Correspondence between virtual addresses and locations established dynamically at run time.
- Binding: process of specifying a location for an address.

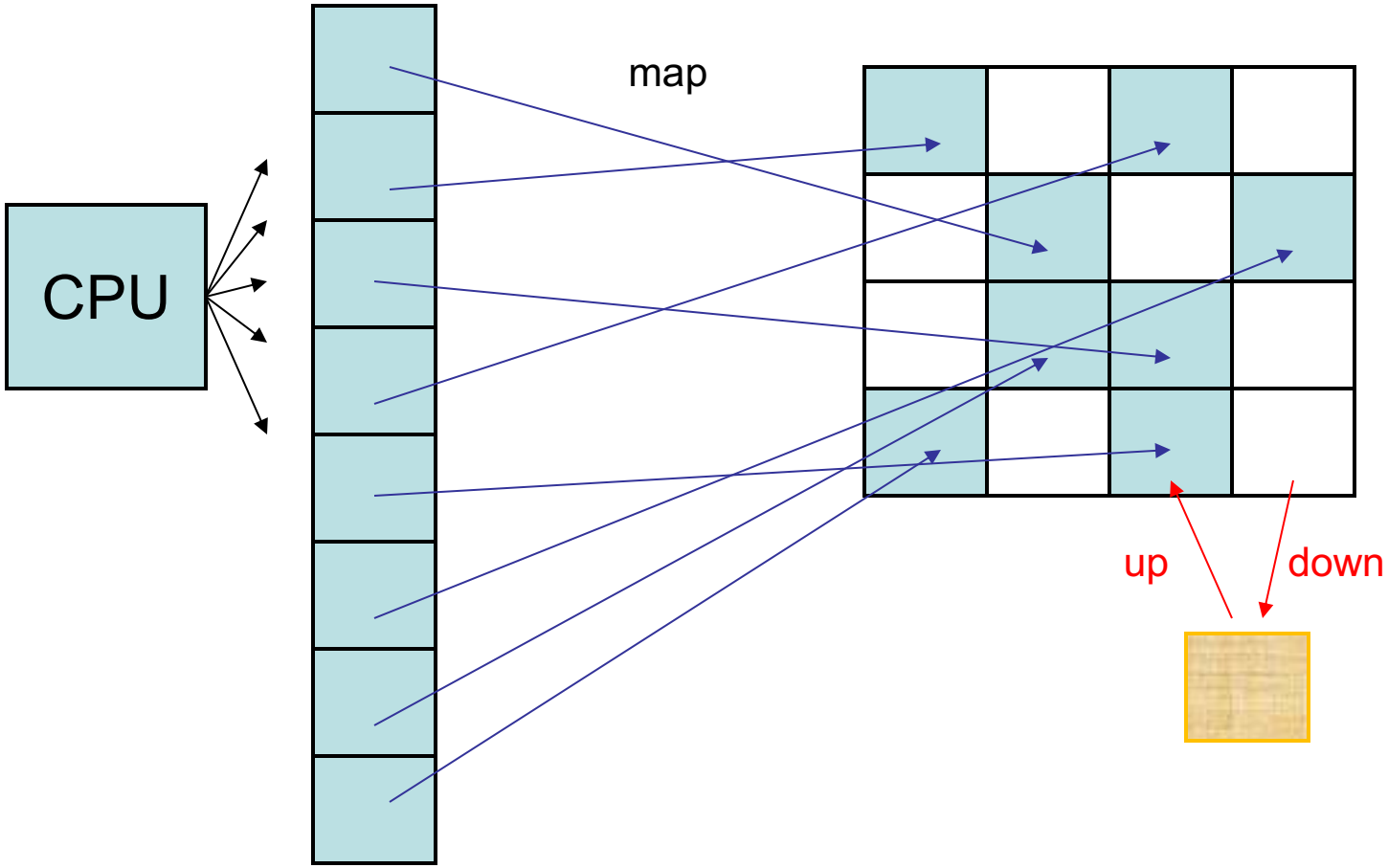
Why Address Spaces?

- Associated uniquely with a process. Contains all the data of the process. Implements the OS guarantee that processes have private, nonoverlapping memories.
- Address space appears to the process as a single RAM all its own.
- The OS simulates the address space using the available RAM by
  - Mapping addresses to locations
  - Moving data up and down hierarchy automatically

Basic Idea of Mapping

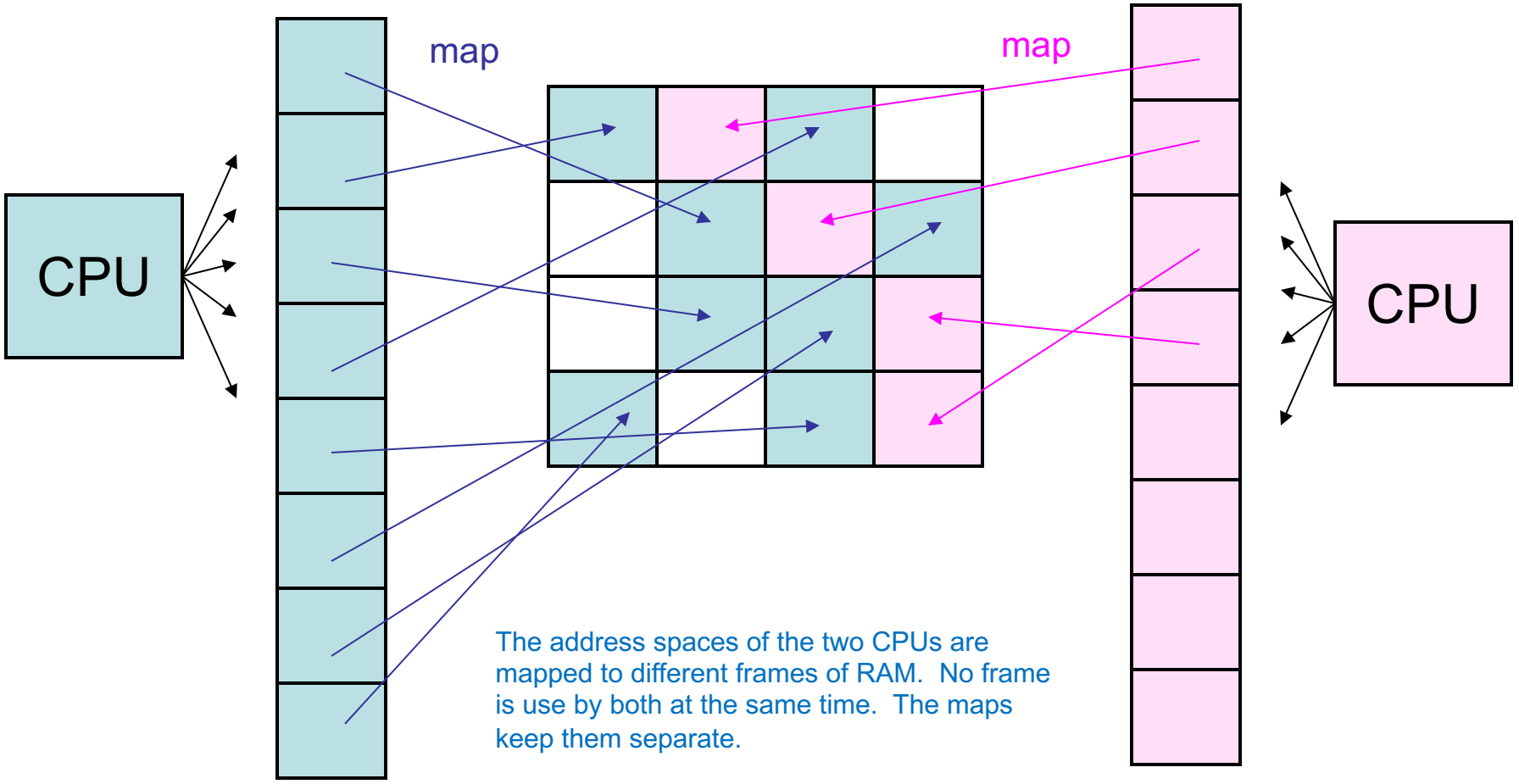


Location Independence



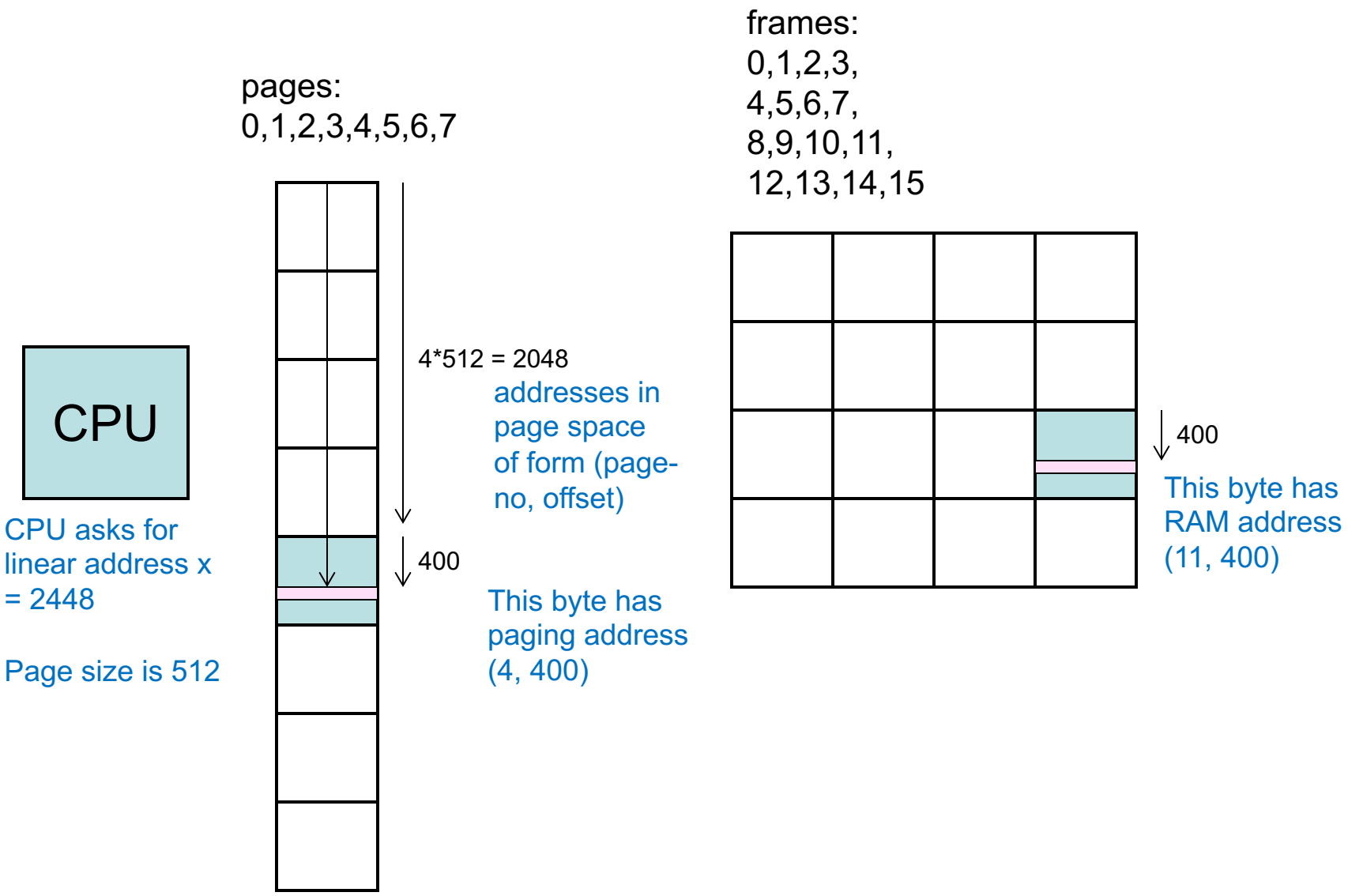
Page 4 was in frame 15 and was moved down to DISK. When it was brought back, frame 15 was no longer available and page 4 went into frame 14 instead. This is invisible to the CPU.

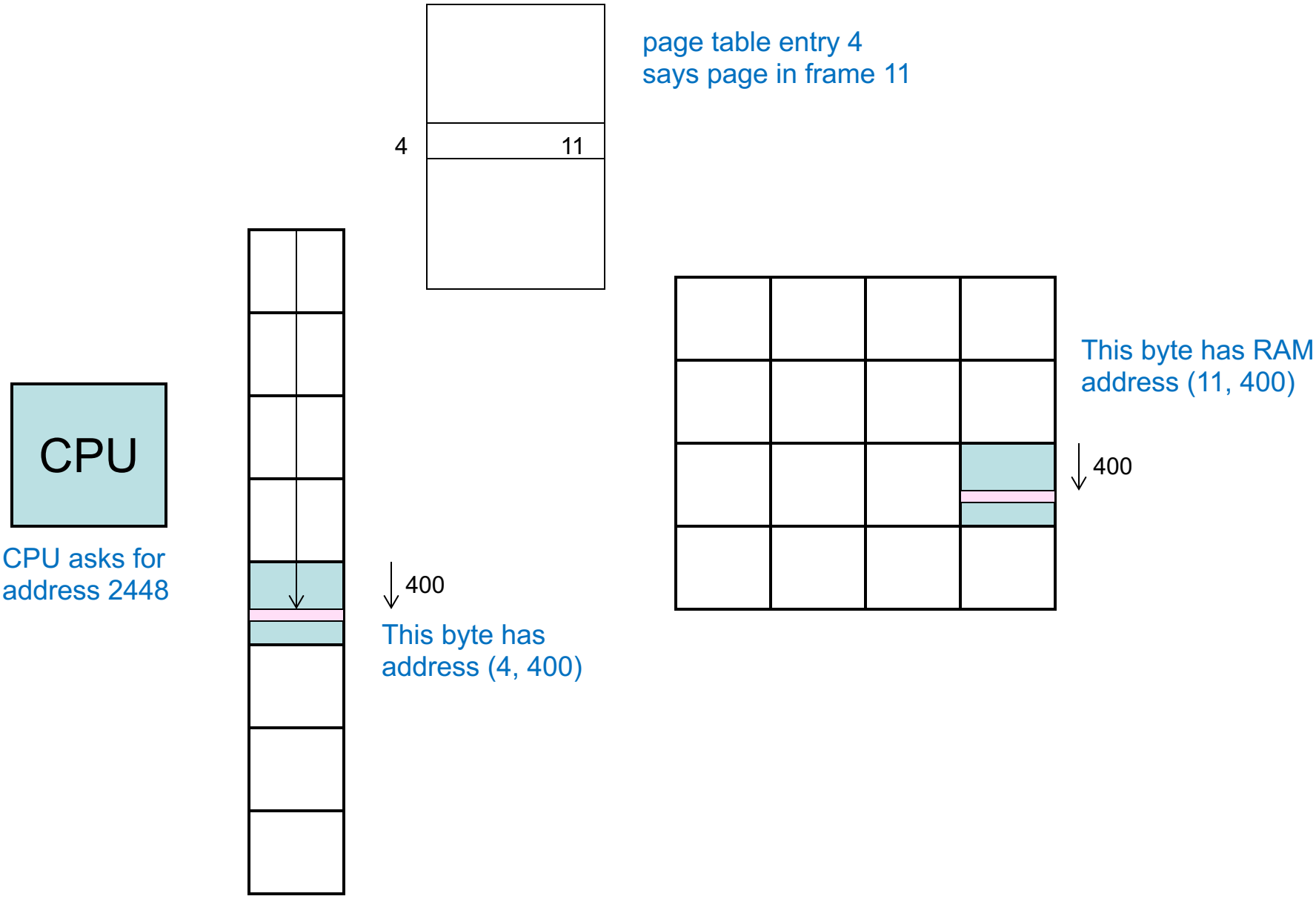
Logical Partition



The address spaces of the two CPUs are mapped to different frames of RAM. No frame is use by both at the same time. The maps keep them separate.

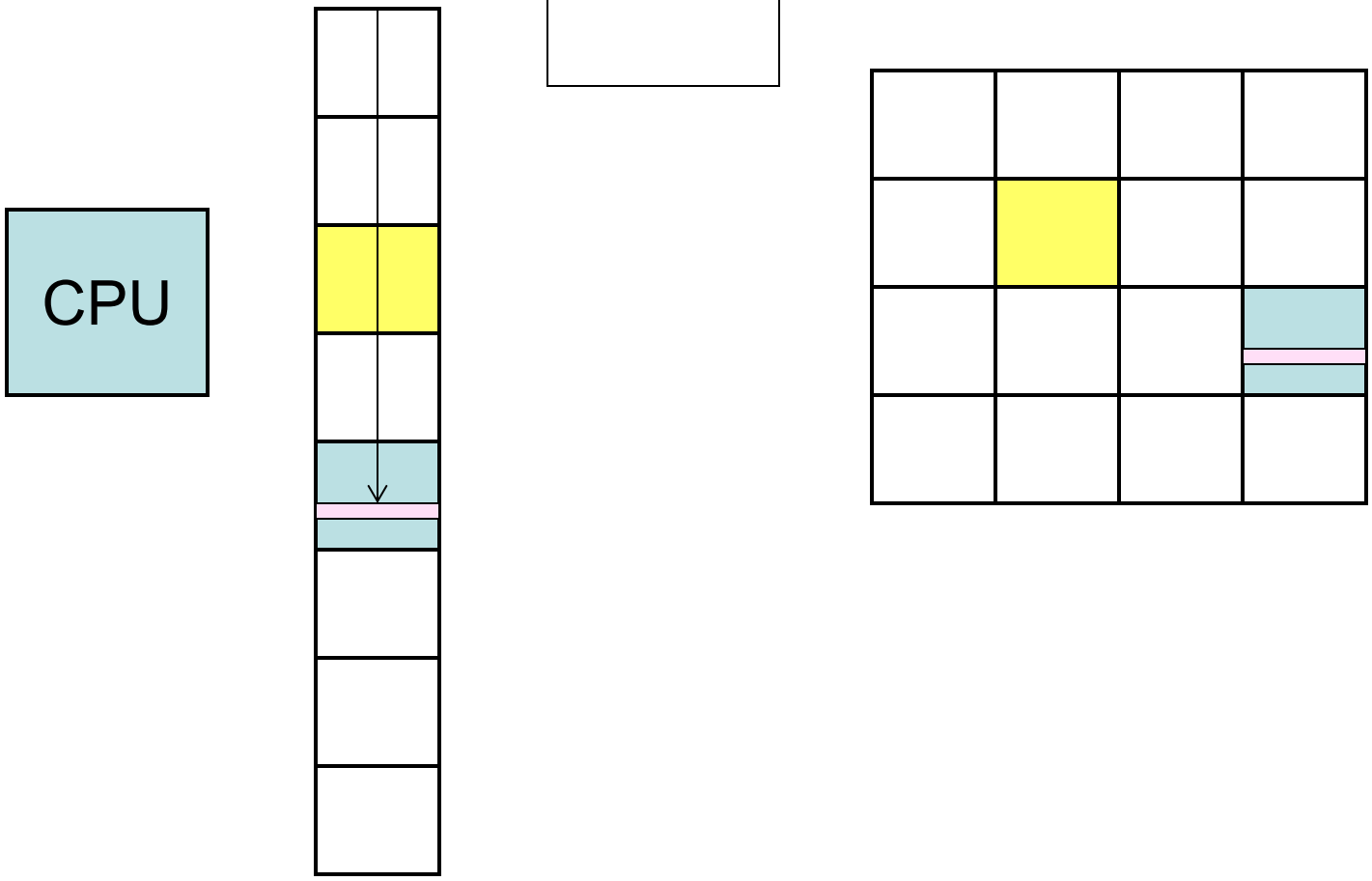
Addressing with pages





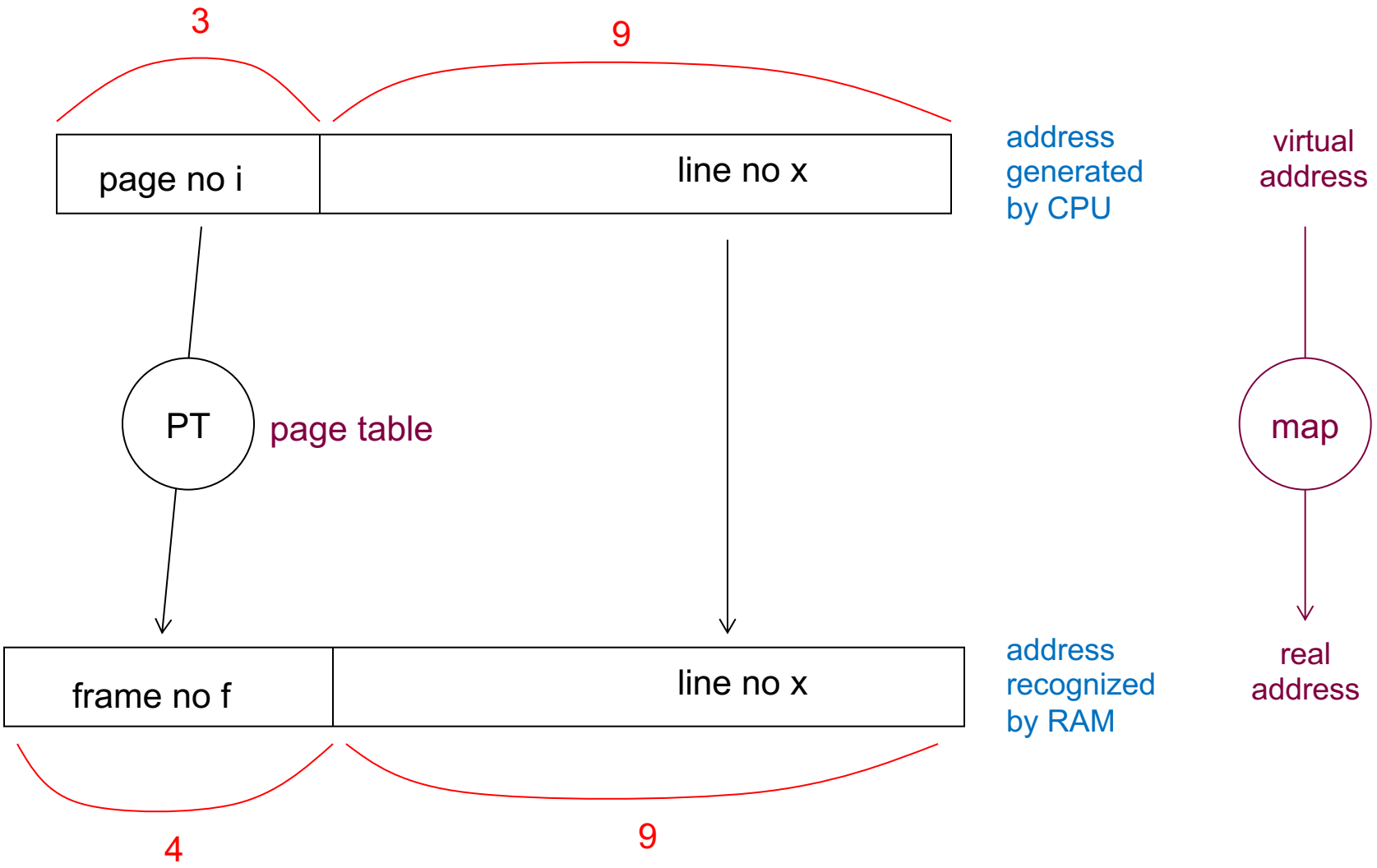
	P	
2	1	5
3	0	
4	1	11

Some pages are not loaded into RAM. A presence bit P in the page table tells which pages are loaded. Here, pages 2 and 4 are present; page 3 is missing.

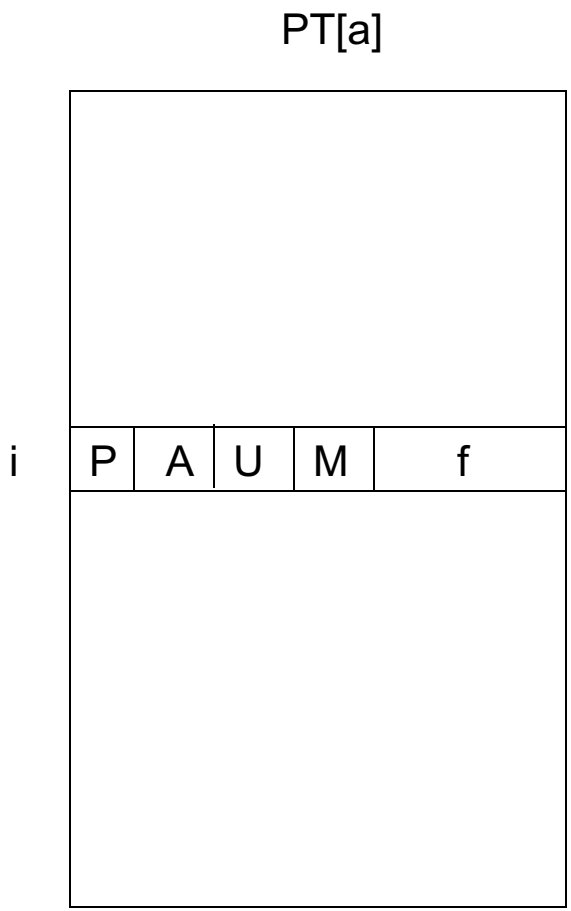


How is the Map Constructed?

- Objective: convert CPU virtual address to RAM frame address
  - CPU address (i,x) where i is 3-bit page number (there are 8 pages) and x is 9-bit line number (there are 512 bytes on a page)
  - RAM address (j,x) where j is 4-bit frame number (there are 16 frames)
  - Here, address space is smaller than RAM. Often address space is larger.



Page Table Format



a = address space number

i = page number

P = presence bit

A = access code

U = used bit

M = modified bit

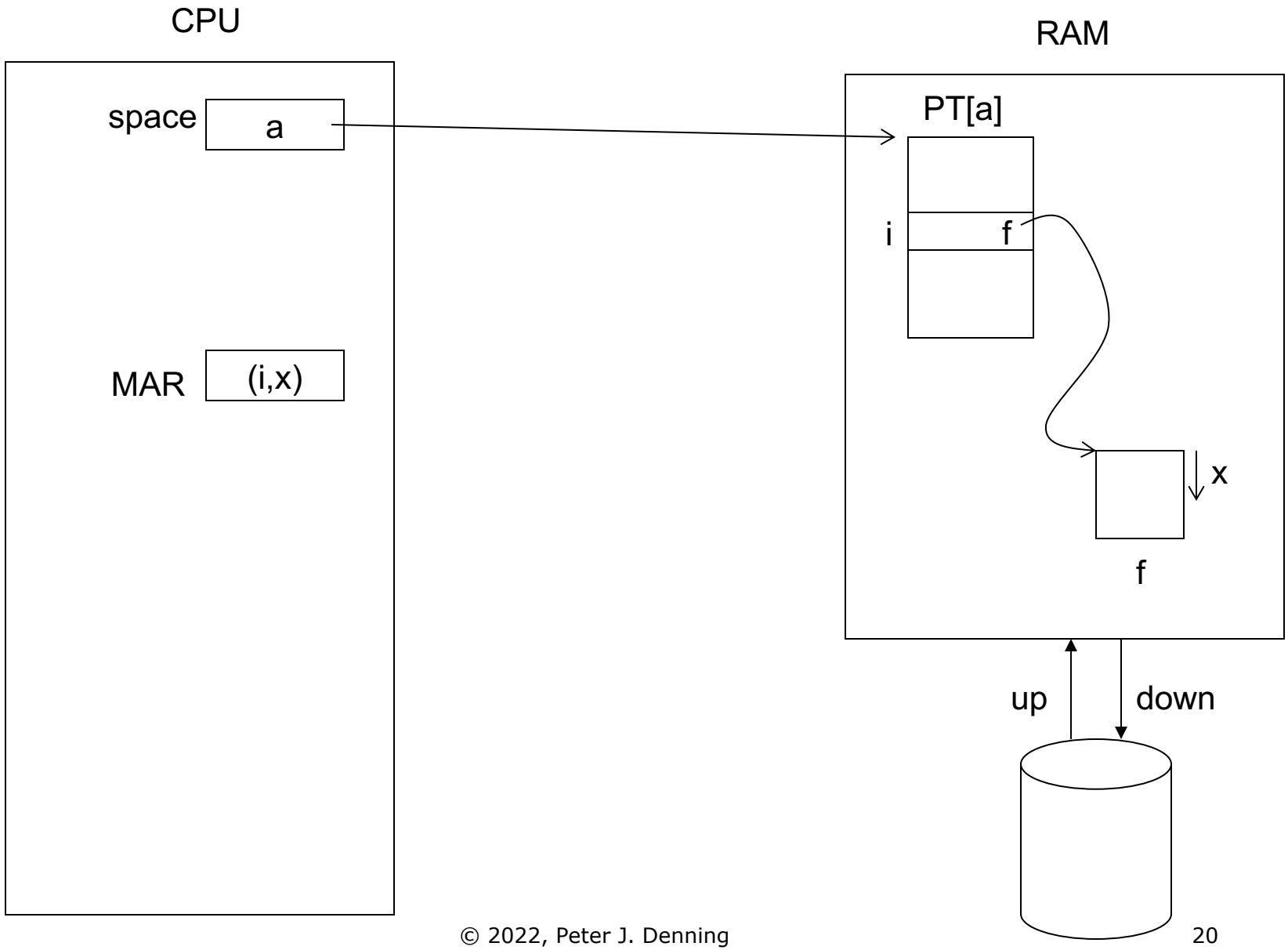
f = frame number

How Big is Page Table?

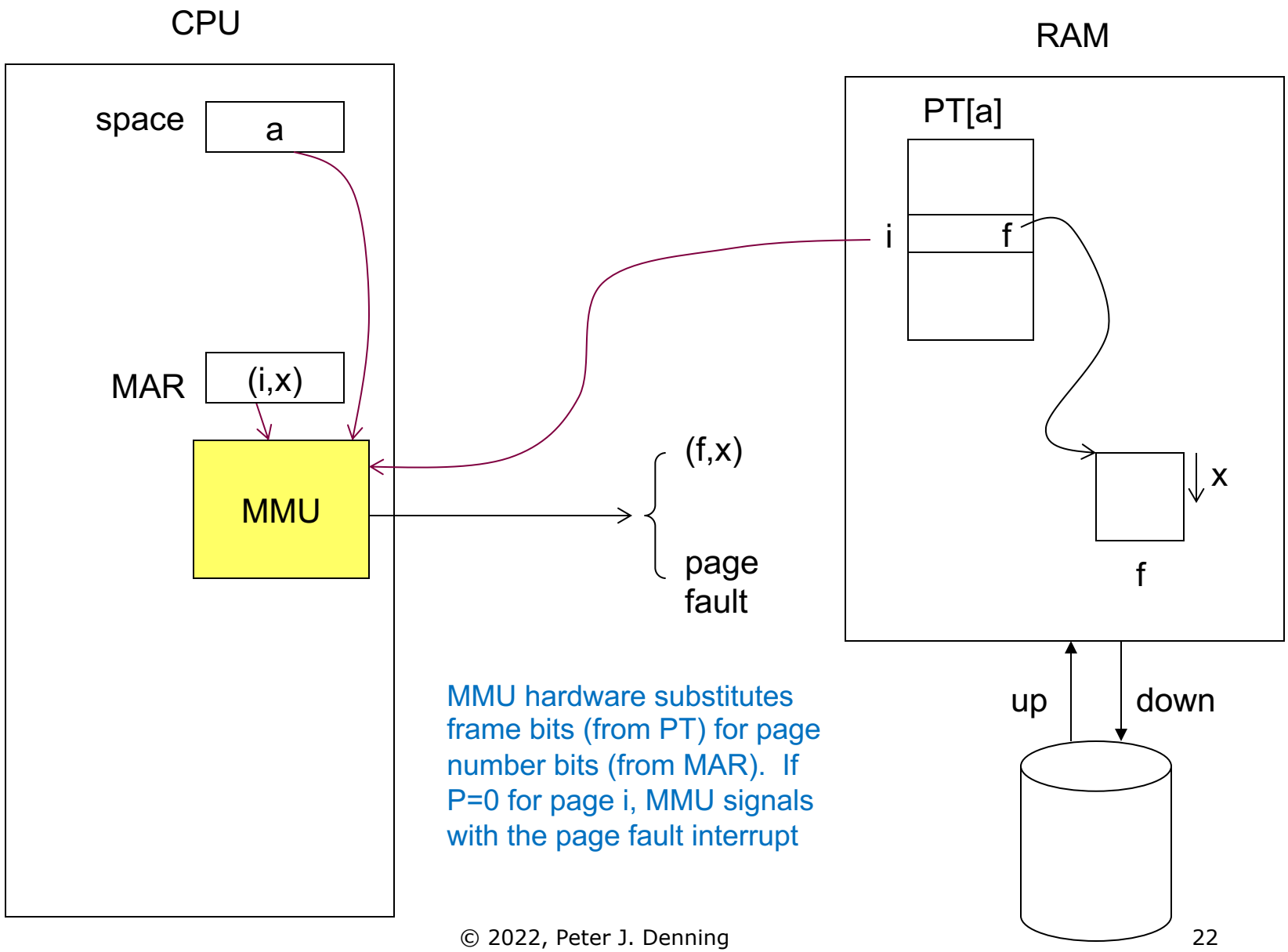
- Consider example where CPU has 16-bit addresses, RAM 32-bit, and page is 1024 bytes (2^{10}).
  - Need 10 bits for the line number, thus 22 bits for frame number ($2^{32}/2^{10} = 2^{22}$)
  - Can fit the P,M,U, and A bits into 1 byte and the frame number into 3 bytes
  - Page table is $4x2^{16} = 2^{18}$ bytes. Tiny fraction of RAM, which is 2^{32} bytes.

How is it all put together?

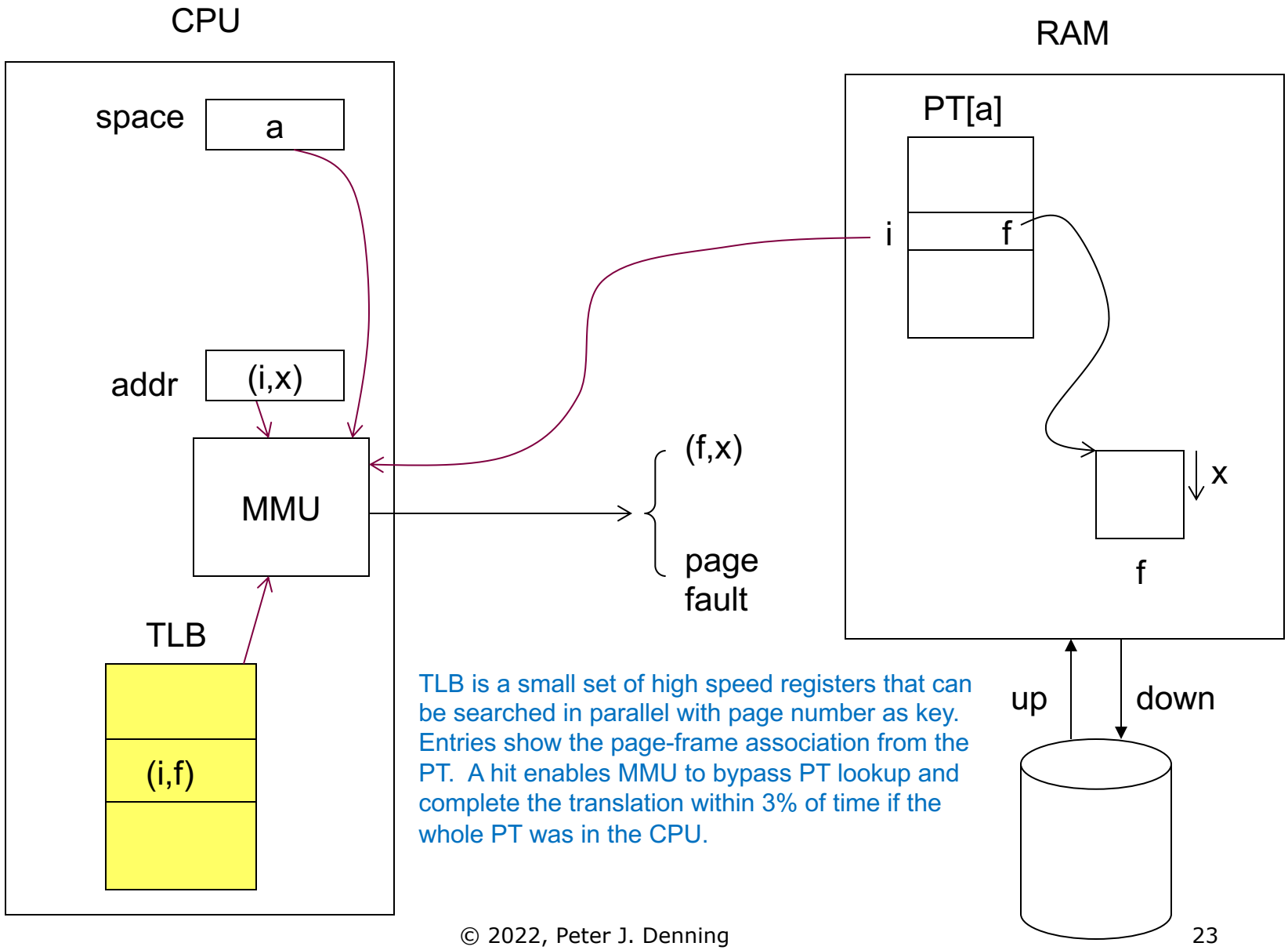
- Next series of pictures shows how all these components are assembled into a working virtual memory system with lightning-fast address mapping.



- CPU generates address (i,x)
- Objective: map to (f,x) in memory
- PT for CPU's address space in kernel area of RAM
- Copy of entire address space in file on disk (the "swap file" or "cache file") -- (Master Copy Principle) -- OS keeps master copy up to date as process writes into some of its pages.

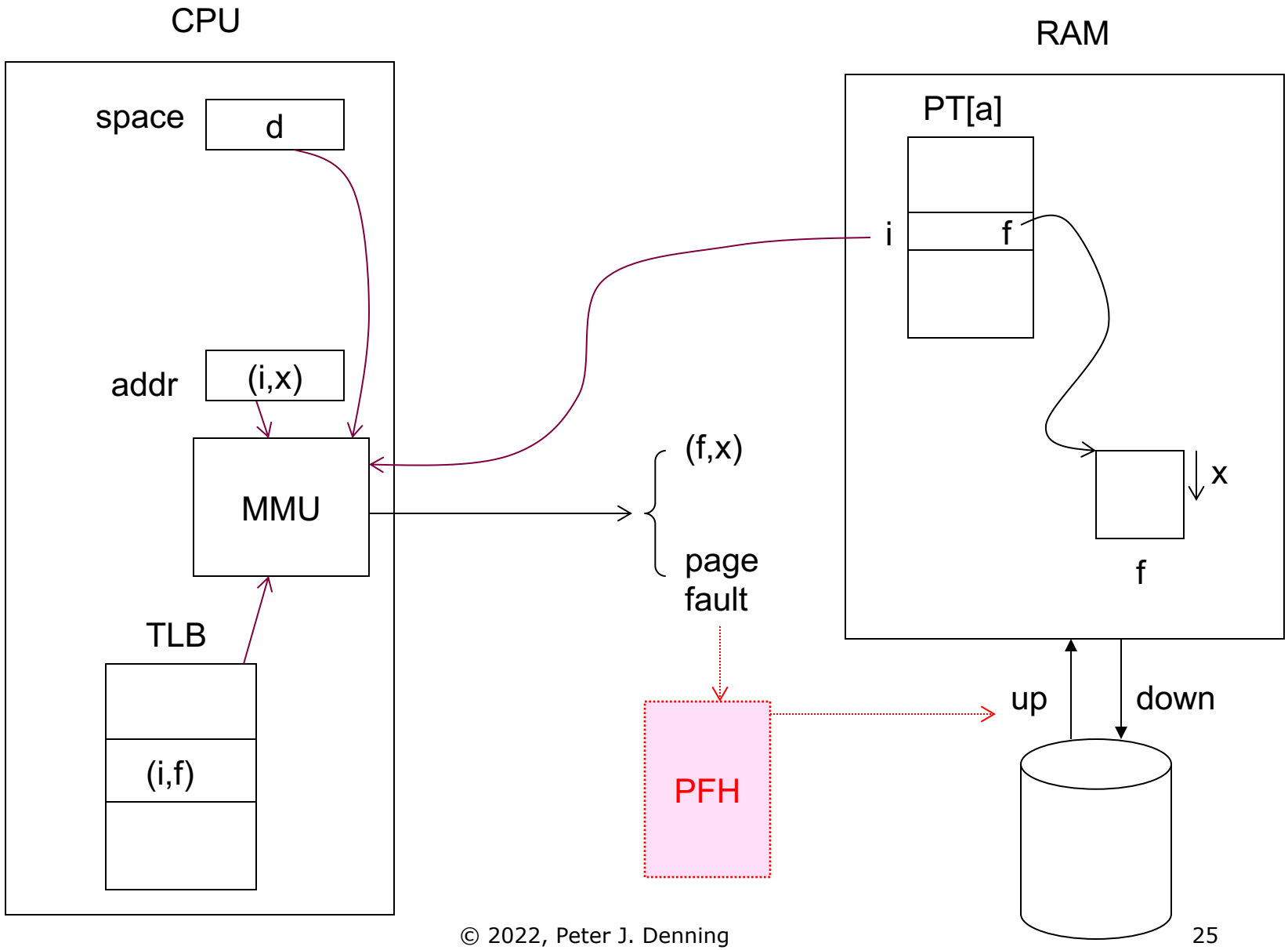


MMU hardware substitutes frame bits (from PT) for page number bits (from MAR). If P=0 for page i, MMU signals with the page fault interrupt



TLB is a small set of high speed registers that can be searched in parallel with page number as key. Entries show the page-frame association from the PT. A hit enables MMU to bypass PT lookup and complete the translation within 3% of time if the whole PT was in the CPU.

- Translation Look Aside Buffer (TLB) is cache of most recent address paths as (i,f) pairs
- Bypass a PT lookup if a “hit” occurs.
- Hit ratio h. Miss ratio (1-h).
- Average mapping time:
  - T = h*TLB + (1-h)*RAM
  - Easy to get below 1.03*RAM with a few hundred TLB entries (3% mapping cost)



Conclusions

- Separating address from location brings powerful benefits including partitioning and relocation.
- Bind addresses to locations with a map that can be changed dynamically.
- Paging map illustrates that mapping function can be built into CPU, where it is very fast and completely invisible to users.