Opportunity

Memory references are dynamically translated into physical addresses at run time

- A process may be swapped in and out of main memory such that it occupies different regions

A process may be broken up into pieces that do not need to located contiguously in main memory

Not all pieces of a process need to be loaded in main memory at once during execution

Execution of a Program

Operating system brings into main memory a few pieces of the program

Resident set - portion of process that is in main memory

An interrupt is generated when an address is needed that is not in main memory – *page fault*

Operating system places the process in a blocking state

Piece of process that contains the logical address is brought into main memory

- Operating system issues a disk I/O Read request
- Another process is dispatched to run while the disk I/O takes place
- An interrupt is issued when disk I/O complete which causes the operating system to place the affected process in the Ready state

Advantages of Breaking up a Process

More processes may be maintained in main memory

- Only load in some of the pieces of each process
- With so many processes in main memory, it is very likely at least one process will be in the Ready/Run state at any particular time

A process may be larger than all of main memory

Terminology

Real memory	physical memory, RAM, main memory
Virtual memory	secondary storage holding process images
Thrashing	phenomenon that a process is spending more time paging than executing
Locality	program and data references within a process tend to cluster; implies that only a few pieces of a process will be needed over a short period of time; possible to make intelligent guesses about which pieces will be needed in the future; suggests that virtual memory may work efficiently

Paging

Each process has its own *page table*

Each page table entry contains the frame number of the corresponding page in main memory

A *resident bit* is needed to indicate whether the page is currently in main memory

Modify bit is needed to indicate if the page has been altered since it was last loaded into main memory

If no change has been made, the page does not have to be written to the disk when it needs to be swapped out

virtual address

Page Number	Offset

page table entry

r	m	. . .	Frame Number

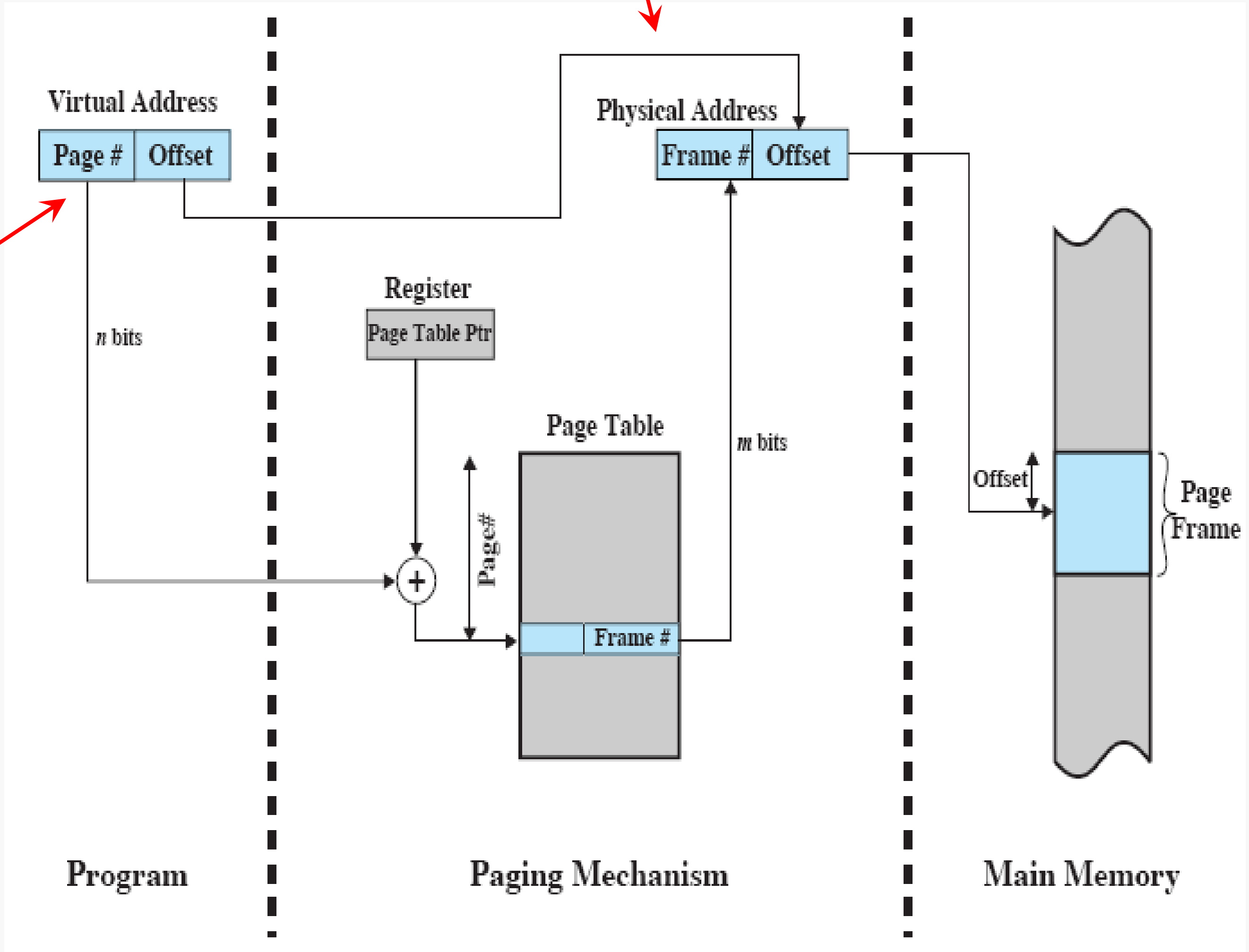
Address Translation

Frame # is just contatenated with offset to obtain physical address

If page size is 2^k…

Page # is leftmost n = 32 - k bits of virtual address…

… so no arithmetic is necessary to extract it…



Cost of Page Indexing

The entire page table may take up too much main memory

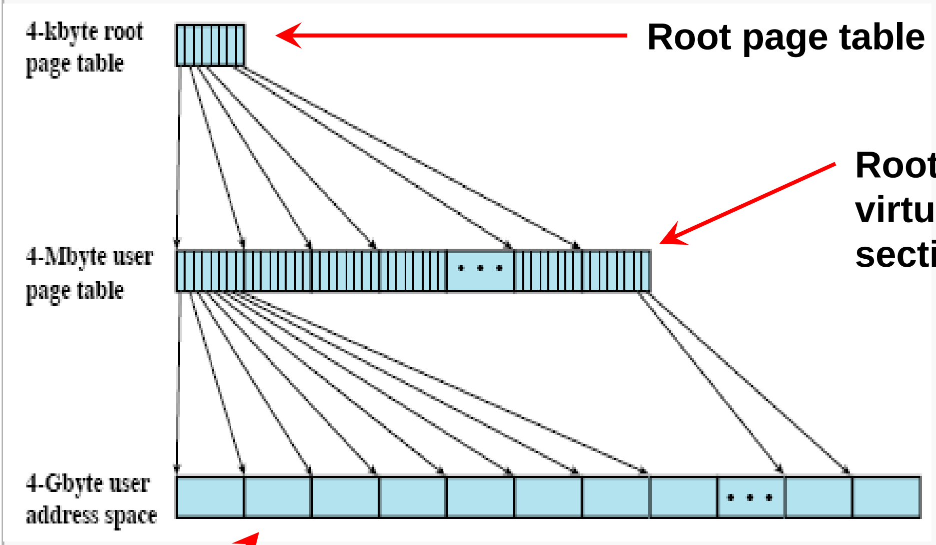
- each process may typically be allowed a virtual memory space of 2 GB or more
- given 4 KB pages, and a 4GB virtual space, there could be 2^{20} page table entries for each process
- each page table entry would occupy, say, 4 bytes of space, so the page table would occupy 4 MB of memory (per process), or 2^{10} frames/pages

Therefore, page tables are also stored in virtual memory and loaded into main memory as needed.

The 2^{20} page table entries can be efficiently indexed using a 2-level structure:

- the 2^{10} pages of the page table can be indexed via a root page table with 2^{10} entries, needing only 4 KB (one page) of main memory
- lock the root page table in main memory…

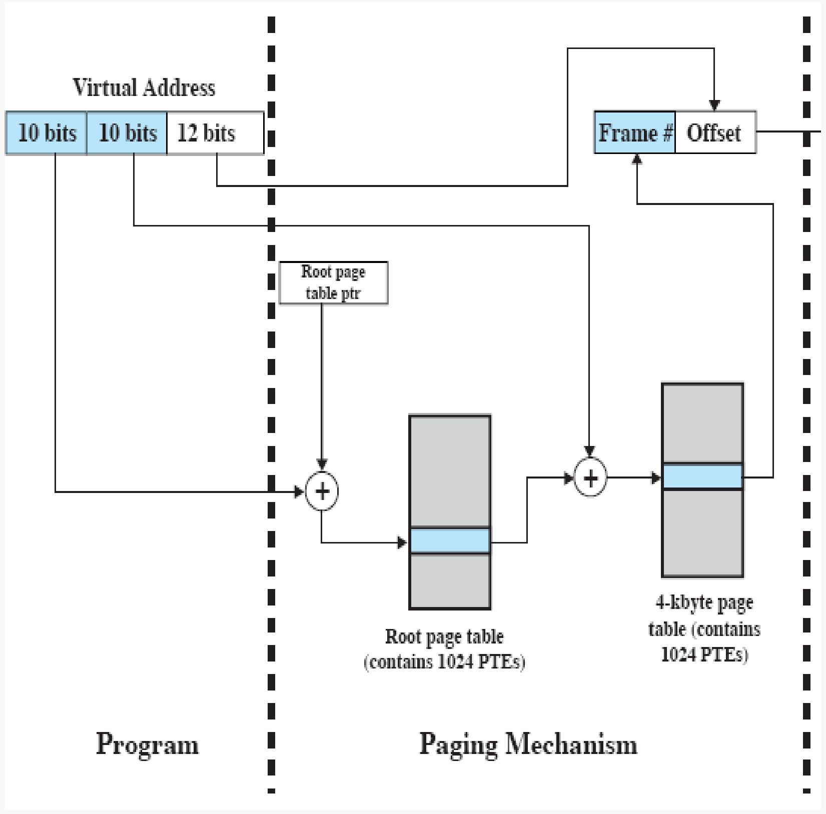
Two-Level Scheme for 32-bit Address



Root page table needs only 1 page of main memory

Root page table provides virtual page # of relevant section of process page table

Each page of the process page table provides access to 2^{10} pages of the process in virtual space



Page Tables

The entire page table may take up too much main memory

Page tables are also stored in virtual memory

When a process is running, part of its page table is in main memory

Inverted page table

- page number portion of a virtual address is mapped into a hash value
- hash value points to inverted page table
- fixed proportion of real memory is required for the tables regardless of the number of processes
- used on PowerPC, UltraSPARC, and IA-64 architecture

Translation Lookaside Buffer

Each virtual memory reference can cause two physical memory accesses

- one to fetch the page table
- one to fetch the data

To overcome this problem a high-speed cache is set up for page table entries

- called a Translation Lookaside Buffer (*TLB*)
- contains page table entries that have been most recently used

Given a virtual address, processor examines the TLB

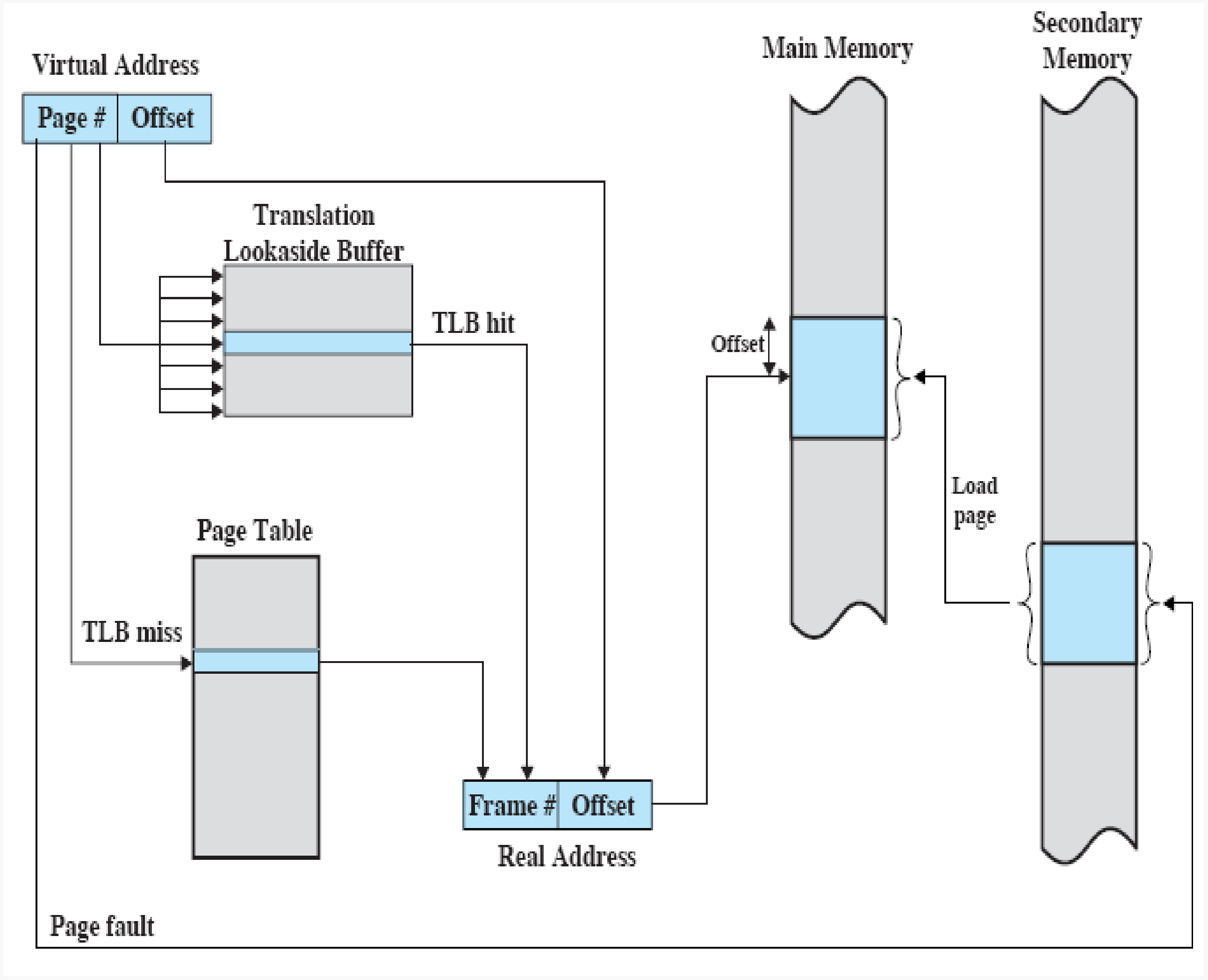
If page table entry is present (*TLB hit*):

- the frame number is retrieved and the real address is formed

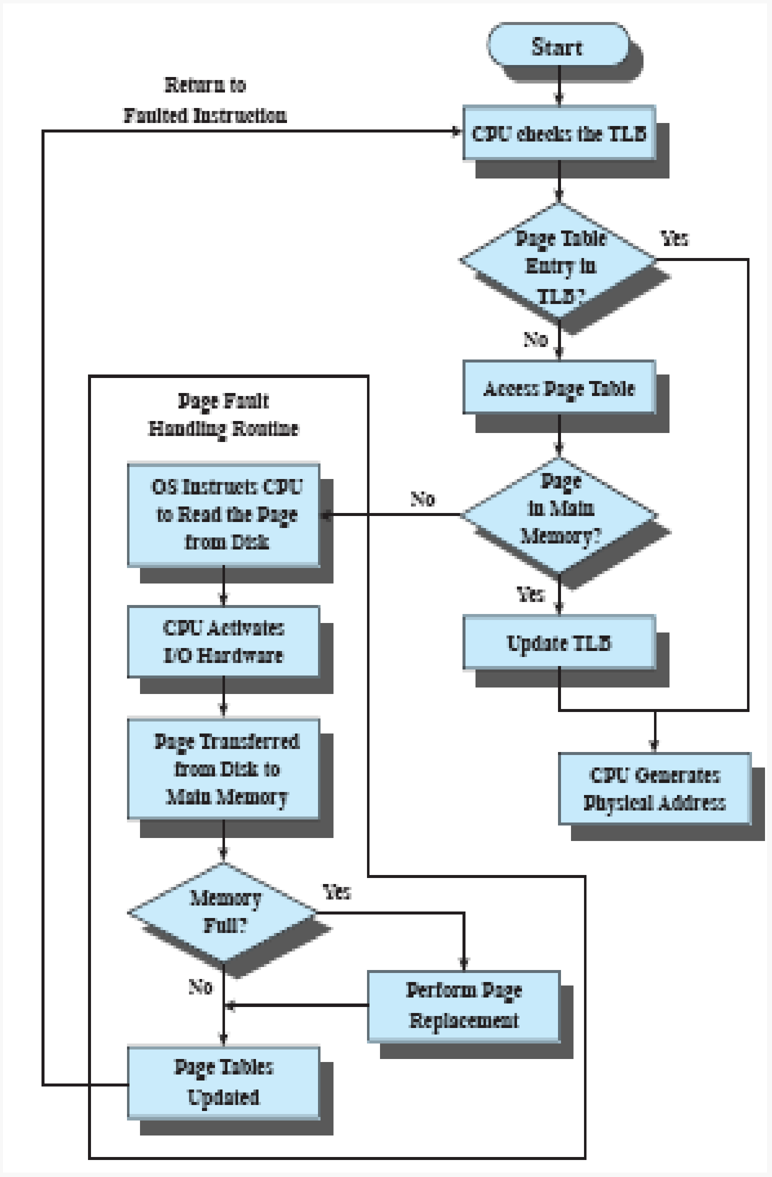
If page table entry is not found in the TLB (*TLB miss*):

- the page number is used to index the process page table
- if page is already in main memory, proceed
- if not, trigger a page fault
- update TLB to index the new page

Translation Lookaside Buffer

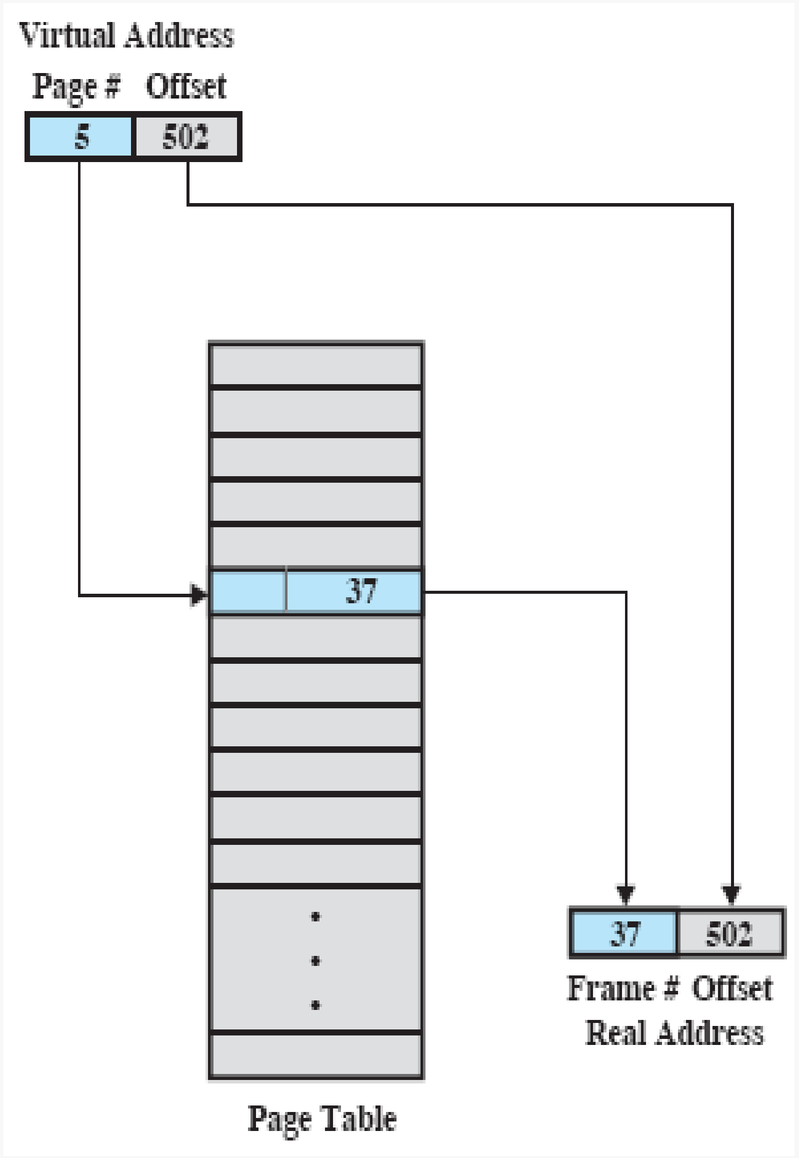


Paging/TLB Flowchart


Start
Return to Faulted Instruction
CPU checks the TLB
Page Table Entry in TLB?
Yes
No
Access Page Table
Page Fault Handling Routine
OS Instructs CPU to Read the Page from Disk
No
Page in Main Memory?
Yes
Update TLB
CPU Activates I/O Hardware
Page Transferred from Disk to Main Memory
CPU Generates Physical Address
Memory Full?
Yes
No
Perform Page Replacement
Page Tables Updated

Direct Mapping

Page # is index of corresponding entry in the page table, so the relevant table entry can be found in Θ(1) time via the page table.



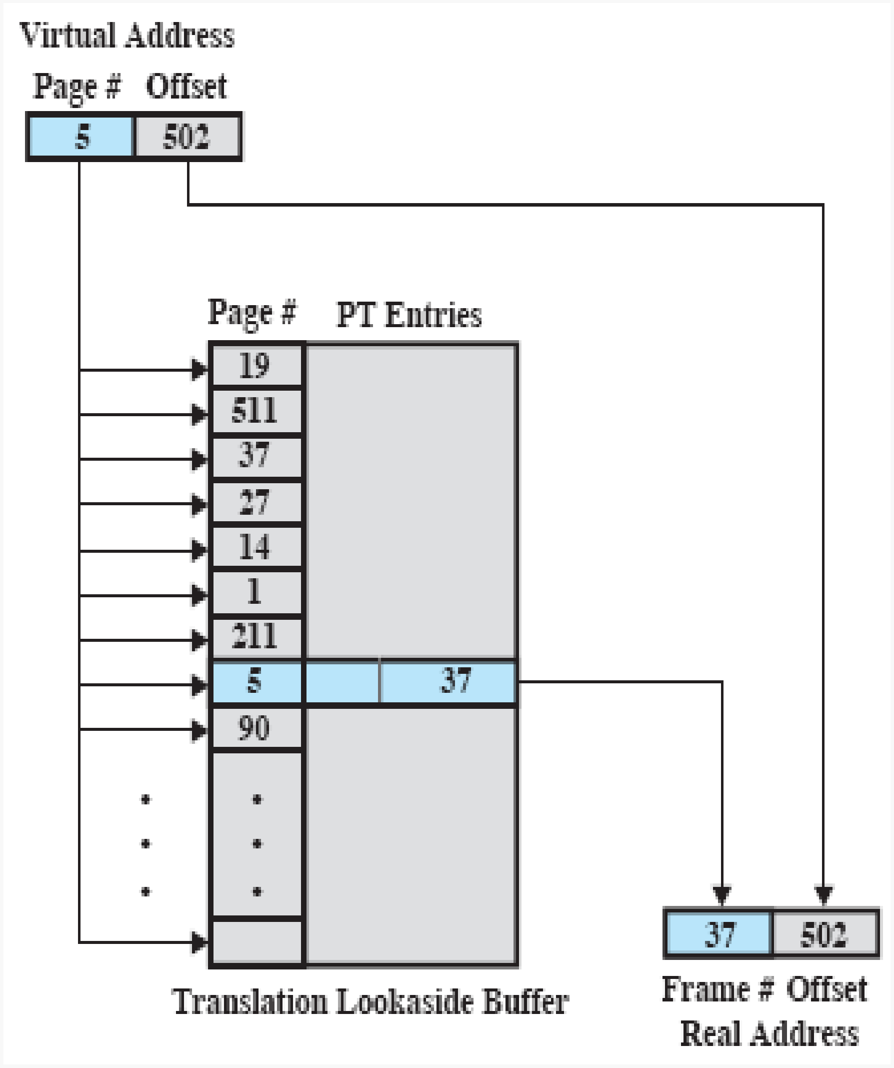
Associative Mapping

Unfortunately, the TLB does not contain all of the entries that would be in the page table.

So, direct mapping won’t work in the TLB.

But, we still need efficient lookup.

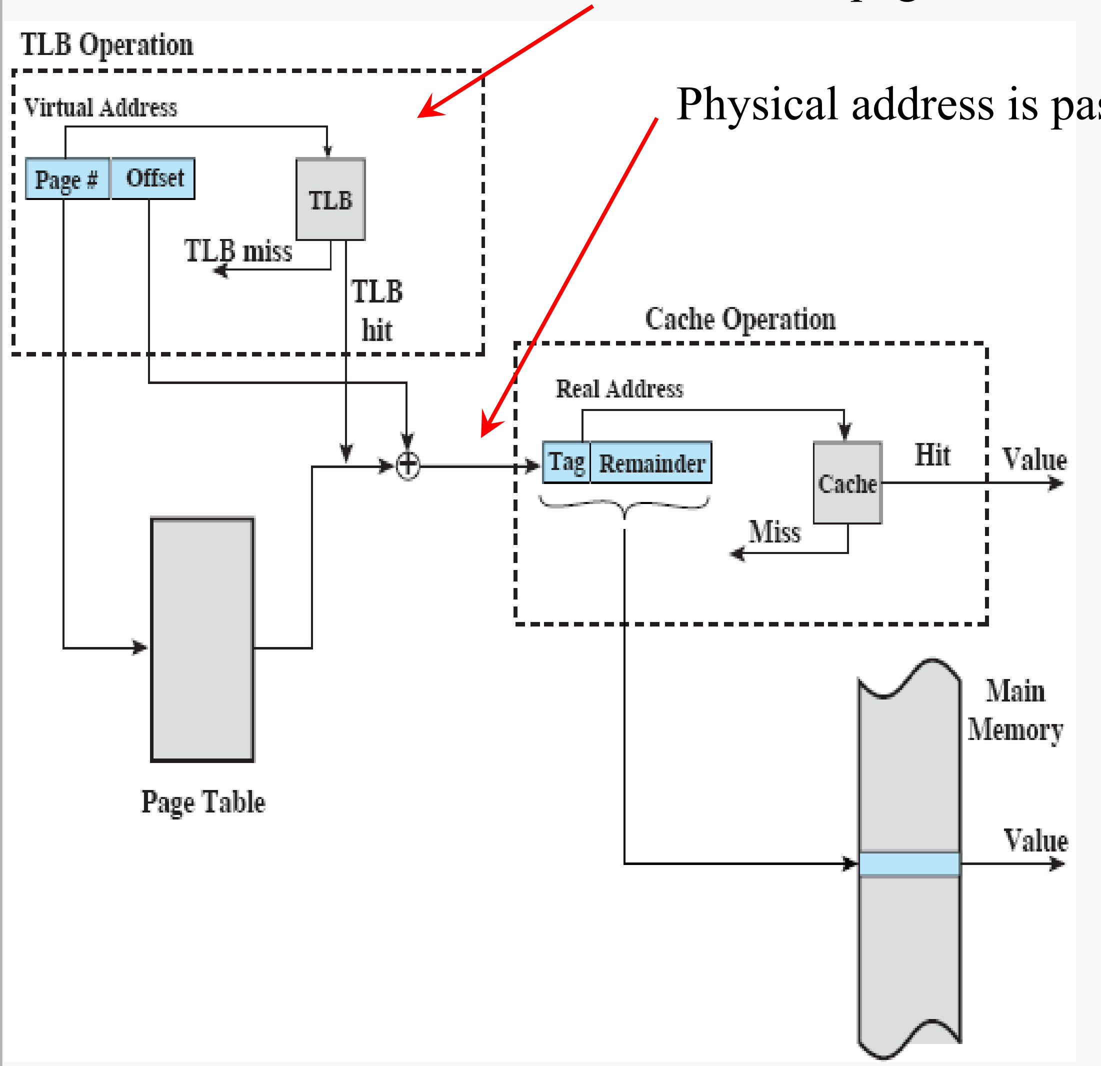
The TLB would support fully associative lookup… in simplest terms, every location in the TLB can be compared to the desired value at once, so lookup will still be Θ(1).



VM/Cache Interaction

Virtual address is resolved into a physical address by using the TLB and page table.

Physical address is passed to cache manager.



If requested location is not in the cache, it is fetched from main memory into the cache and then served up to the process.