23. Virtual Memory: Making VM fast

EECS 370 – Introduction to Computer Organization – Fall 2015

Profs. Dreslinski, Mudge, and Wenisch

EECS Department
University of Michigan in Ann Arbor, USA

Announcements

TODAY is the last lecture with new material!

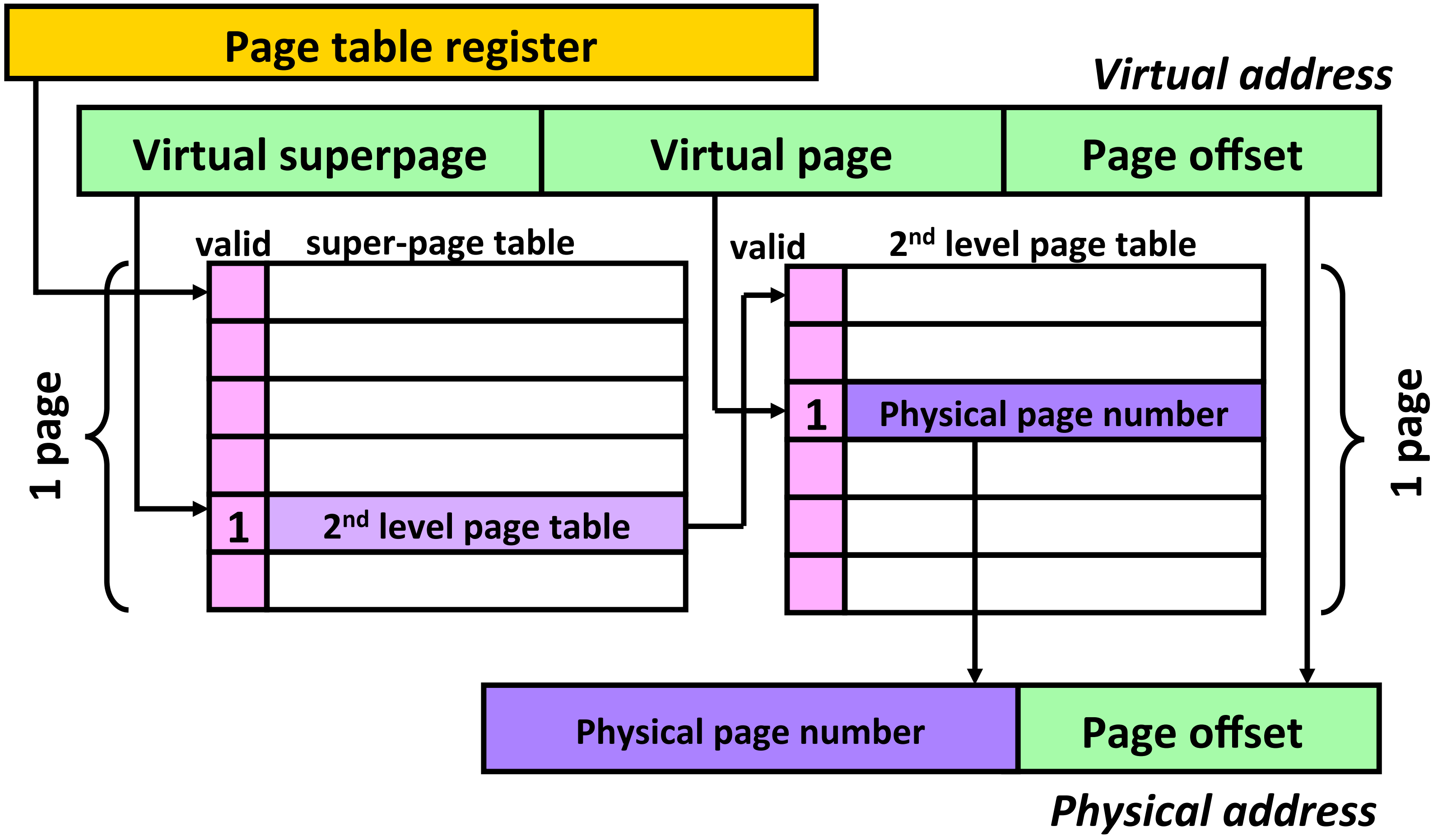
- No class next Tuesday, 12/8—MICRO Conference
- Reviews in all 3 lecture sections next Thursday, 12/10
  - No Prof. office hours
- Project 4 due Thursday, 12/10
- HW 6 due Friday, 12/11

Review: virtual memory

- Virtual memory lets the programmer "see" a memory array larger than the DRAM available on a particular computer system.

- Virtual memory enables multiple programs to share the physical memory without
  - Knowing other programs exist (**transparency**) or
  - Worrying about one program modifying the data contents of another (**protection**).

Review: hierarchical page table



Performance of virtual memory

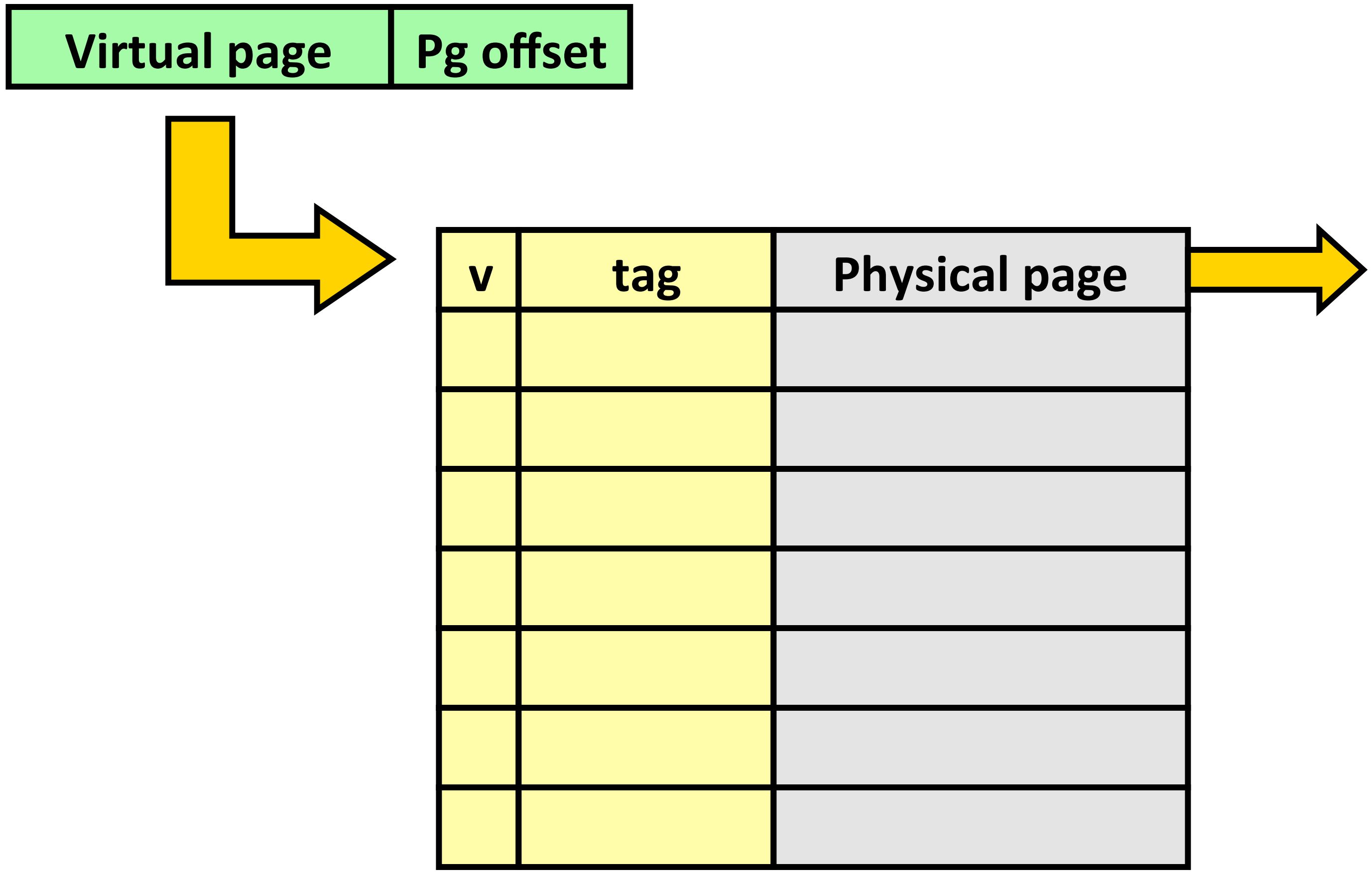
- To translate a virtual address into a physical address, we must first access the page table in physical memory.

- Then we access physical memory again to get the data
  - A load instruction performs at least 2 memory reads.
  - A store instruction performs at least 1 read and then a write.
- What if we were to read from main memory when doing the page table lookup? Slow.

Translation look-aside buffer

- We fix this performance problem by avoiding main memory in the translation from virtual to physical pages.
- Buffer common translations in a **Translation Look-aside Buffer (TLB)**, a fast cache memory dedicated to storing a small subset of valid V-to-P translations.
- 16-512 entries common.
- Generally has low miss rate (< 1%).

TLB



Where is the TLB lookup?

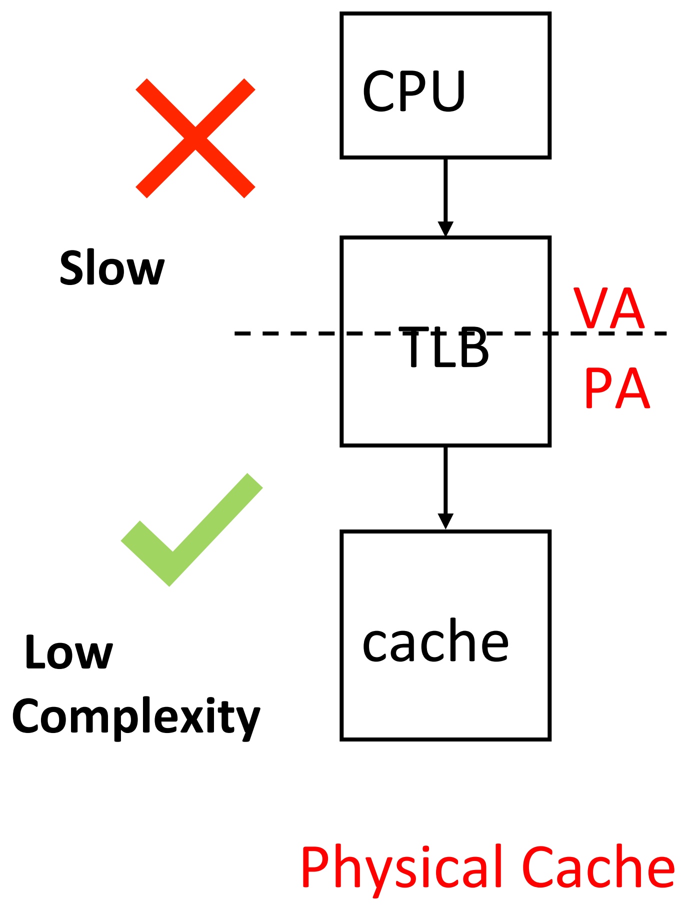
- We put the TLB lookup in the pipeline after the virtual address is calculated and before the memory reference is performed.
  - This may be before or during the data cache access.
  - Without a TLB hit we need to perform the translation during the memory stage of the pipeline.

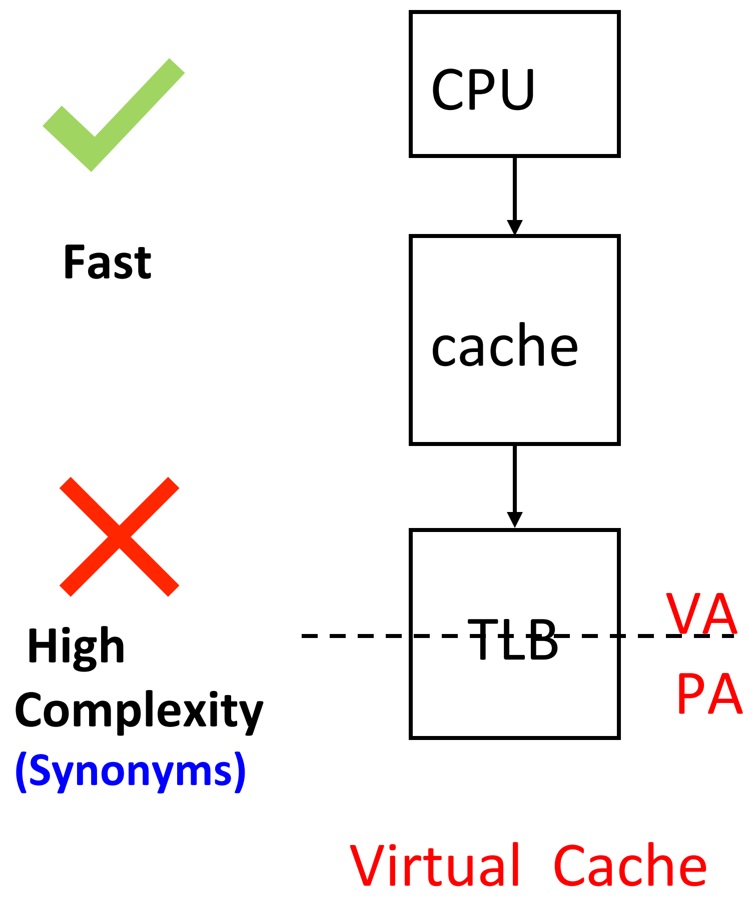
Next topic: Caches in Systems with VM

- VM systems give us two different addresses: virtual and physical.

- Which address should we use to access the data cache?
  - Physical address (after VM translations).
    - Delayed access.
  - Virtual address (before VM translation).
    - Faster access.
    - More complex.

Cache & VM Organization

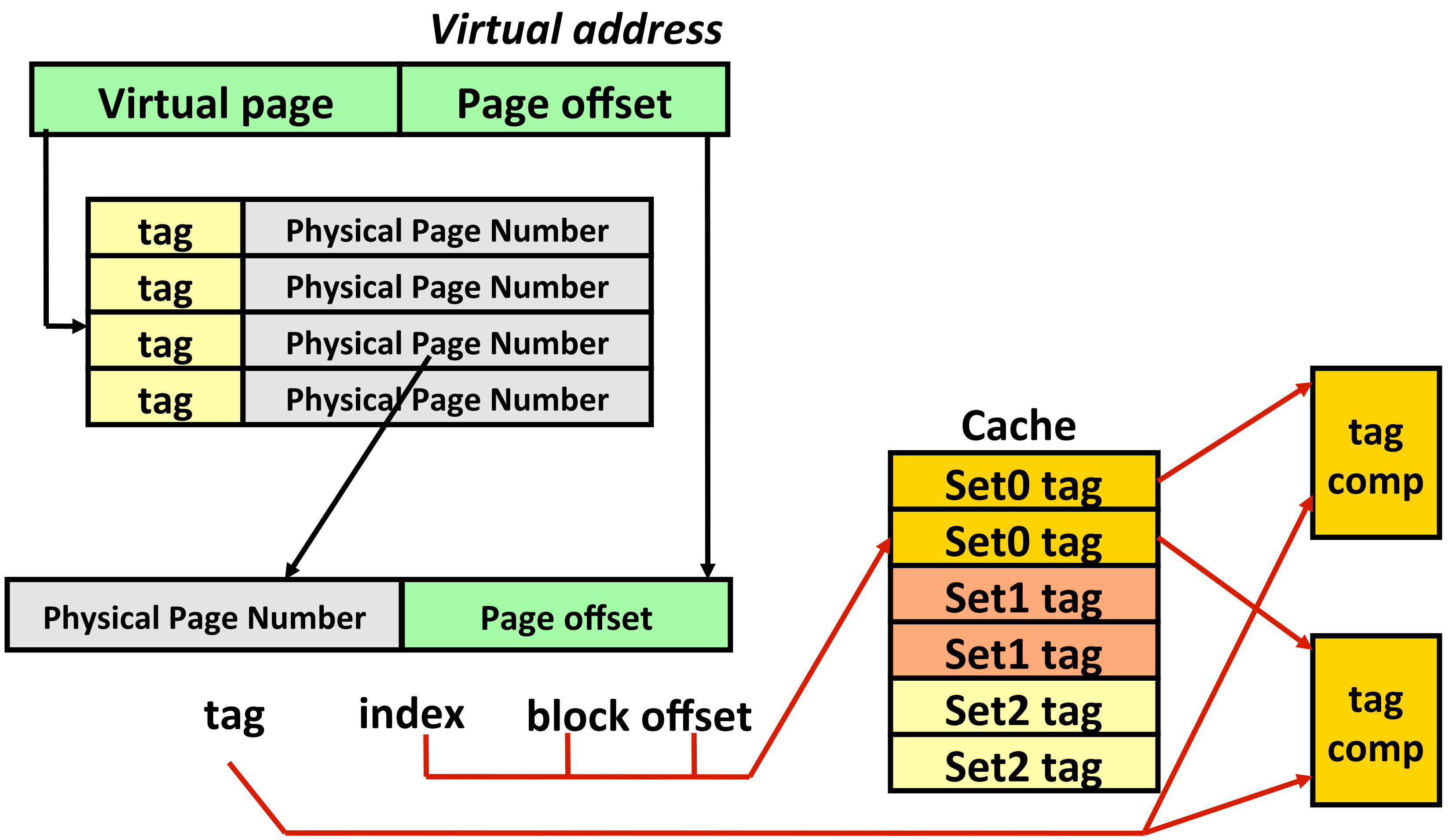




Physically addressed caches

- Perform TLB lookup ***before*** cache tag comparison.
  - Use bits from physical address to index set.
  - Use bits from physical address to compare tag.
- Slower access?
  - Tag lookup takes place ***after*** the TLB lookup.
- Simplifies some VM management.
  - When switching processes, TLB must be invalidated, but cache OK to stay as is.
  - Implications? Might result in fewer cache misses if context switches very common (but they generally are not).

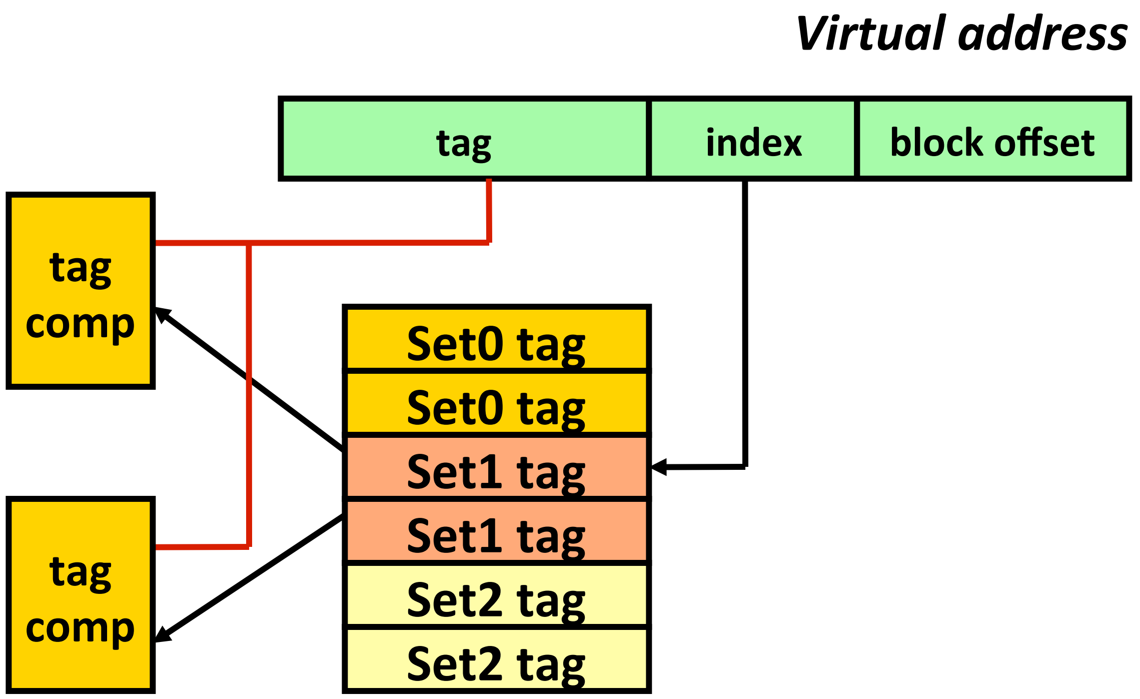
Physically addressed caches



Virtually addressed caches

- Perform the TLB lookup at the same time as the cache tag compare.
  - Uses bits from the virtual address to index the cache set
  - Uses bits from the virtual address for tag match.
- Problems:
  - Aliasing: Two processes may refer to the same physical location with different virtual addresses.
  - When switching processes, TLB must be invalidated, dirty cache blocks must be written back to memory, and cache must be invalidated.

Virtually addressed caches

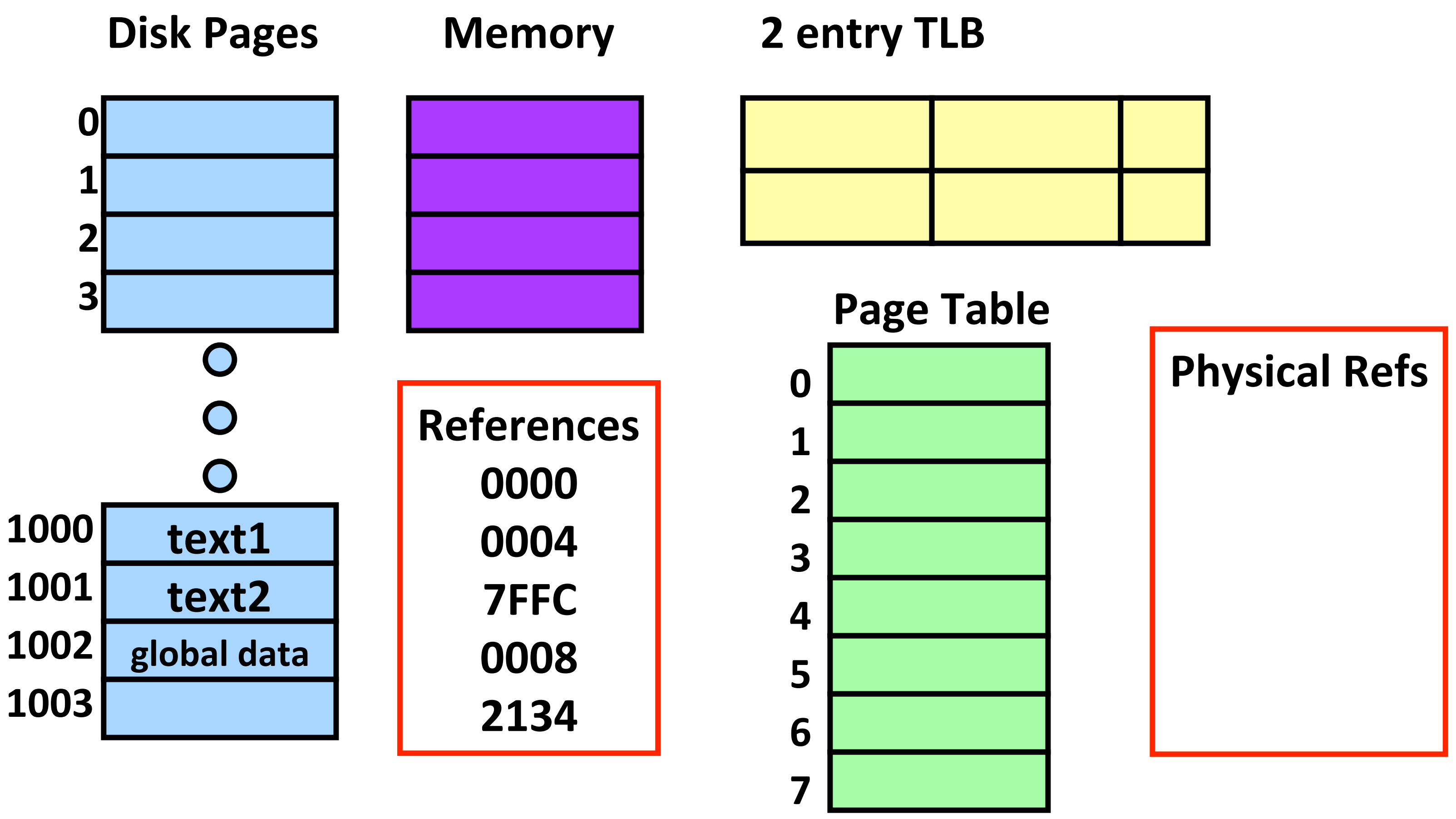


- TLB is accessed in parallel with cache lookup.
- Physical address is used to access main memory in case of a cache miss.

OS support for virtual memory

- It must be able to modify the page table register, update page table values, etc.
- To enable the OS to do this, **BUT** not the user program, we have different execution modes for a process.
  - Executive (or supervisor or kernel level) permissions and
  - User level permissions.

Loading a program into memory

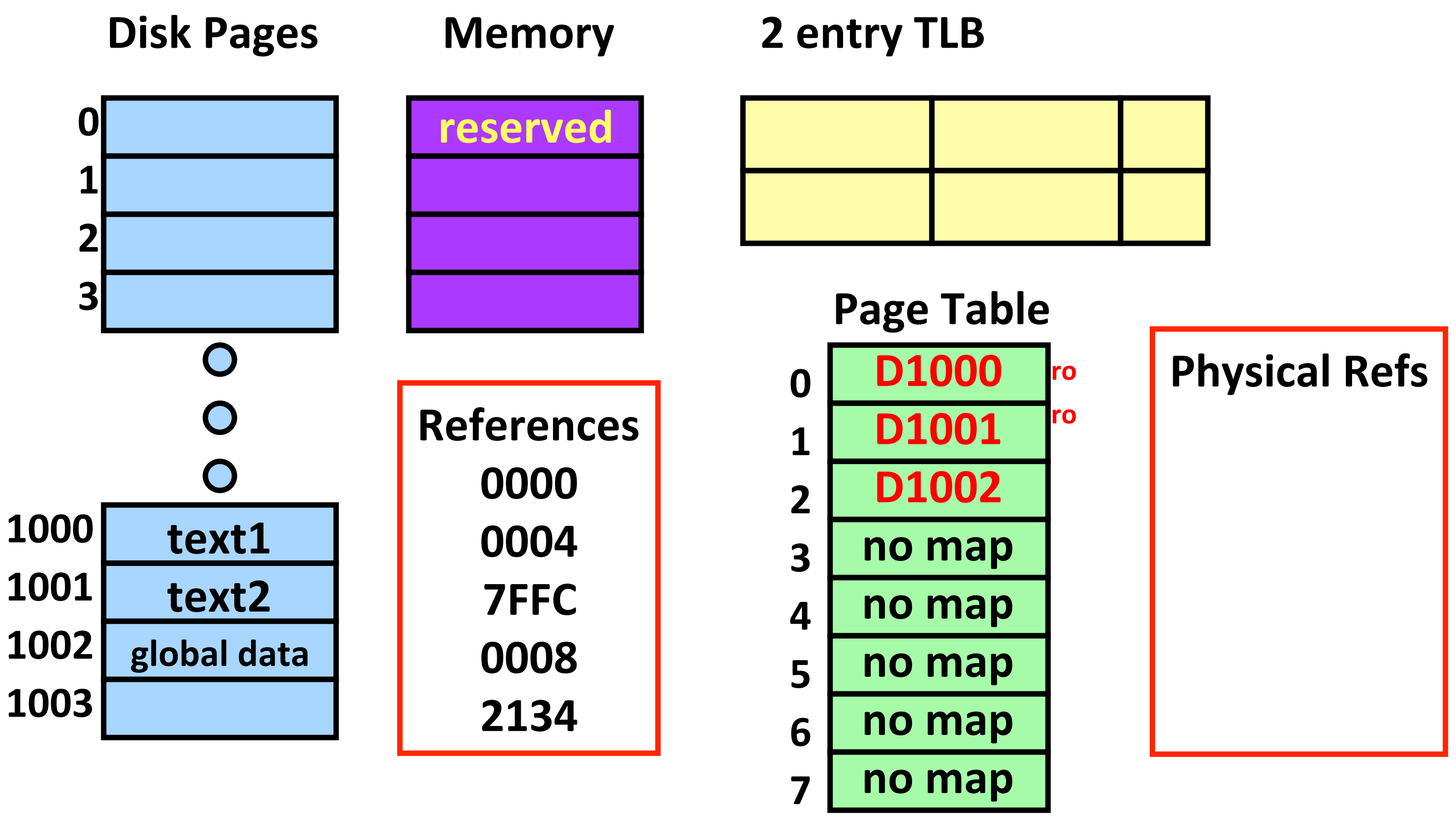


References
0000
0004
7FFC
0008
2134

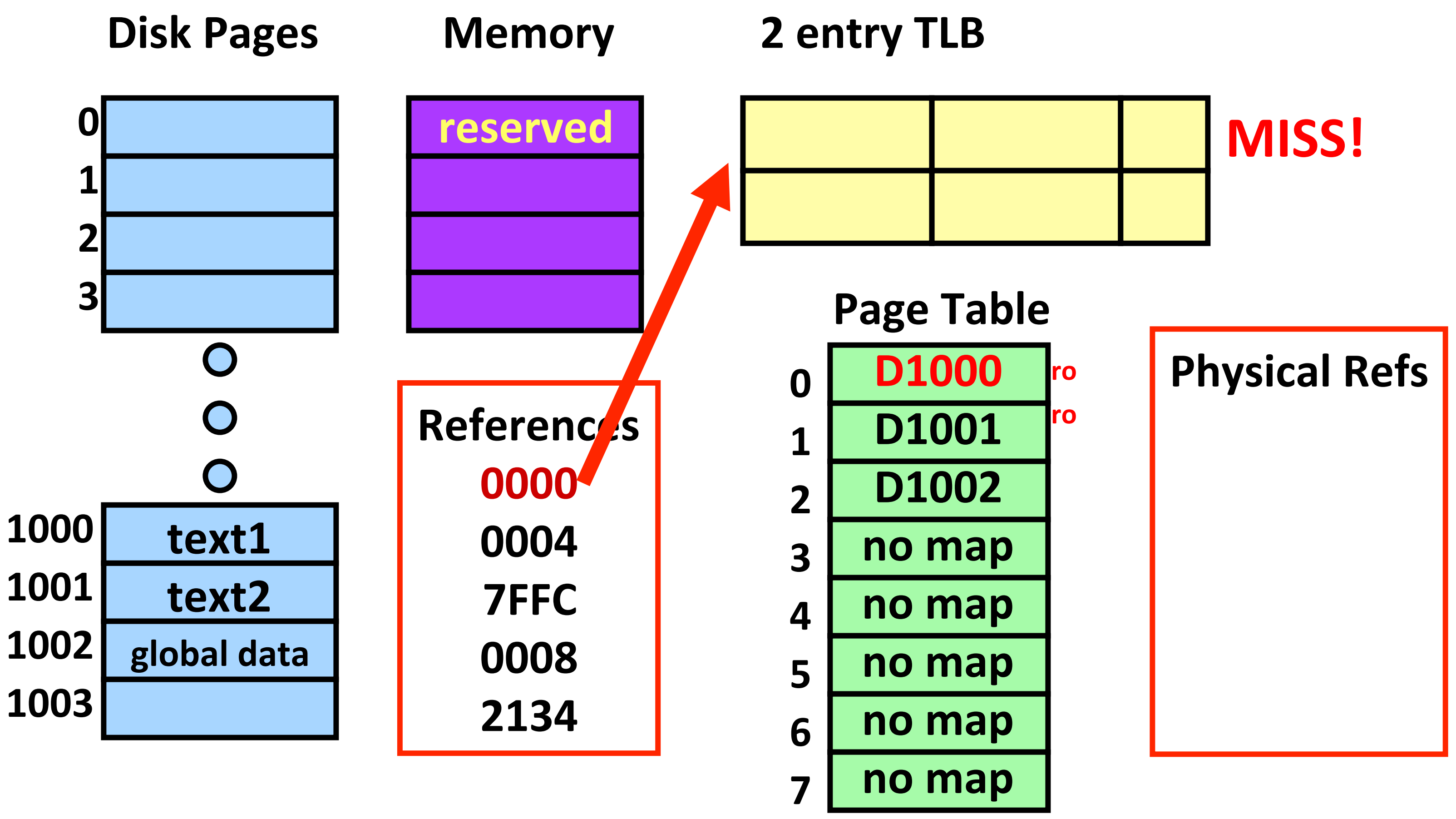
Additional information

- Page size = 4 KB.
- Page table entry size = 4 B.
- Page table register points to physical address 0000.

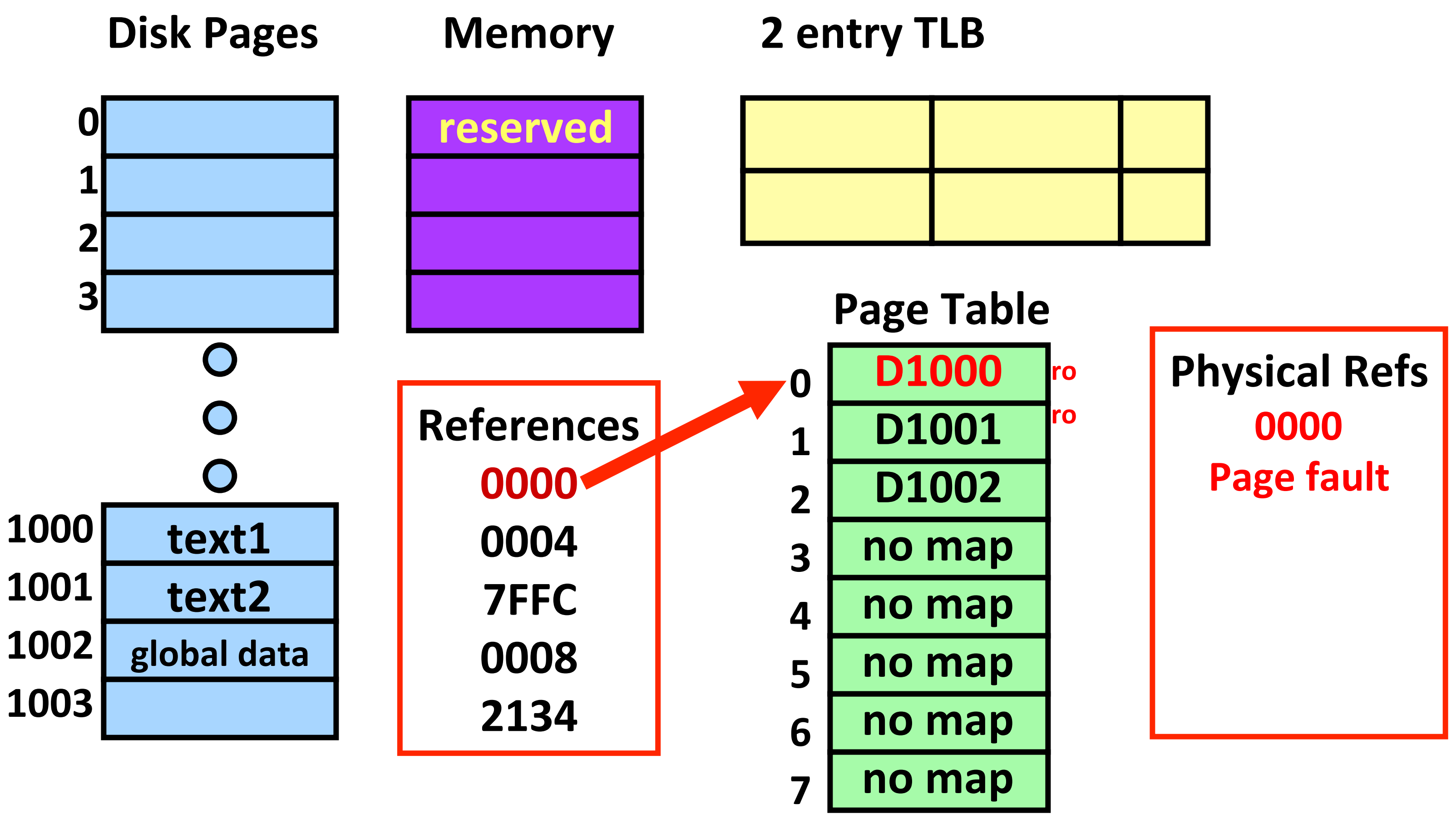
Step 1: read executable header and initialize page table



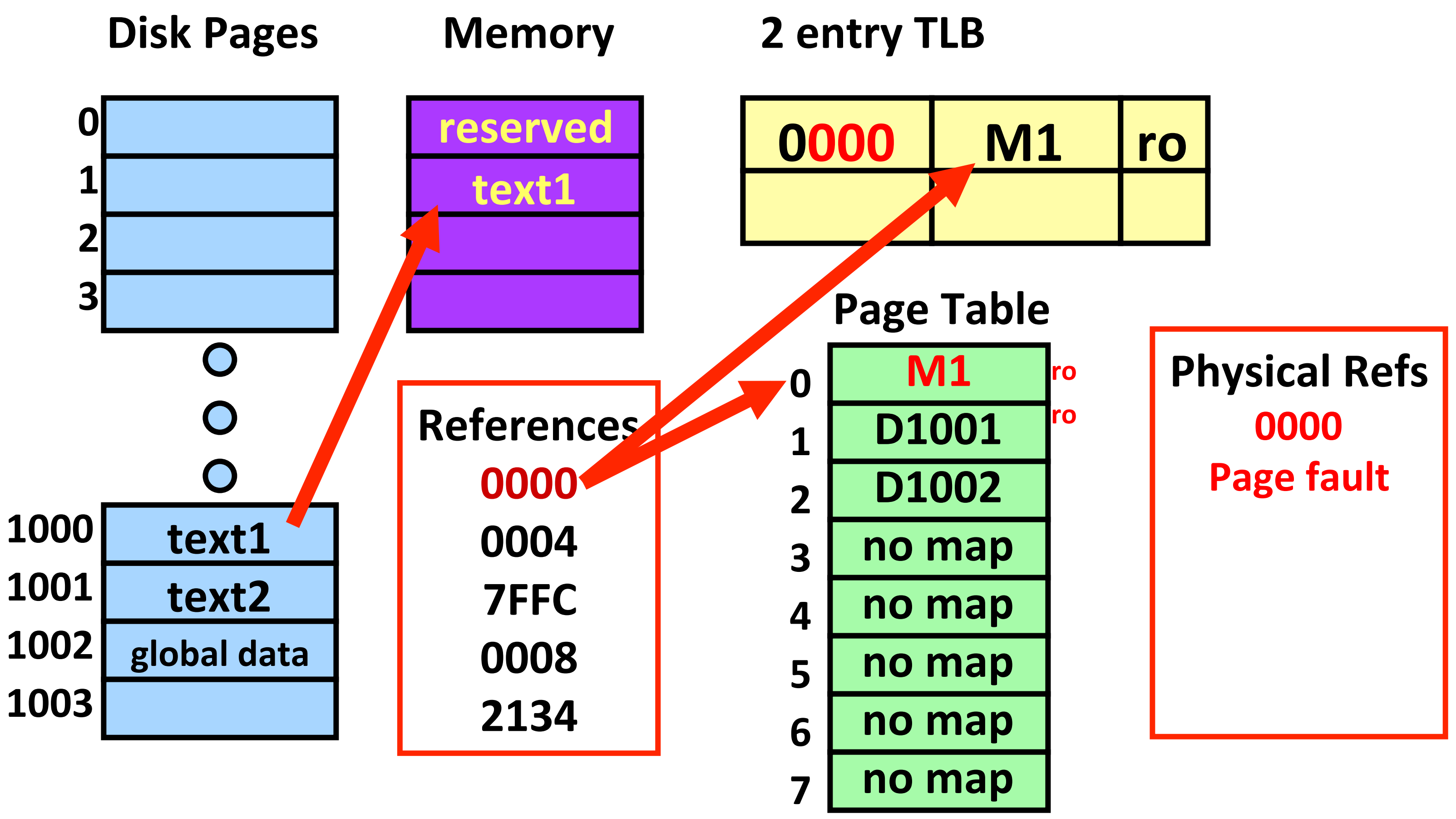
Step 2: load PC from header and start execution



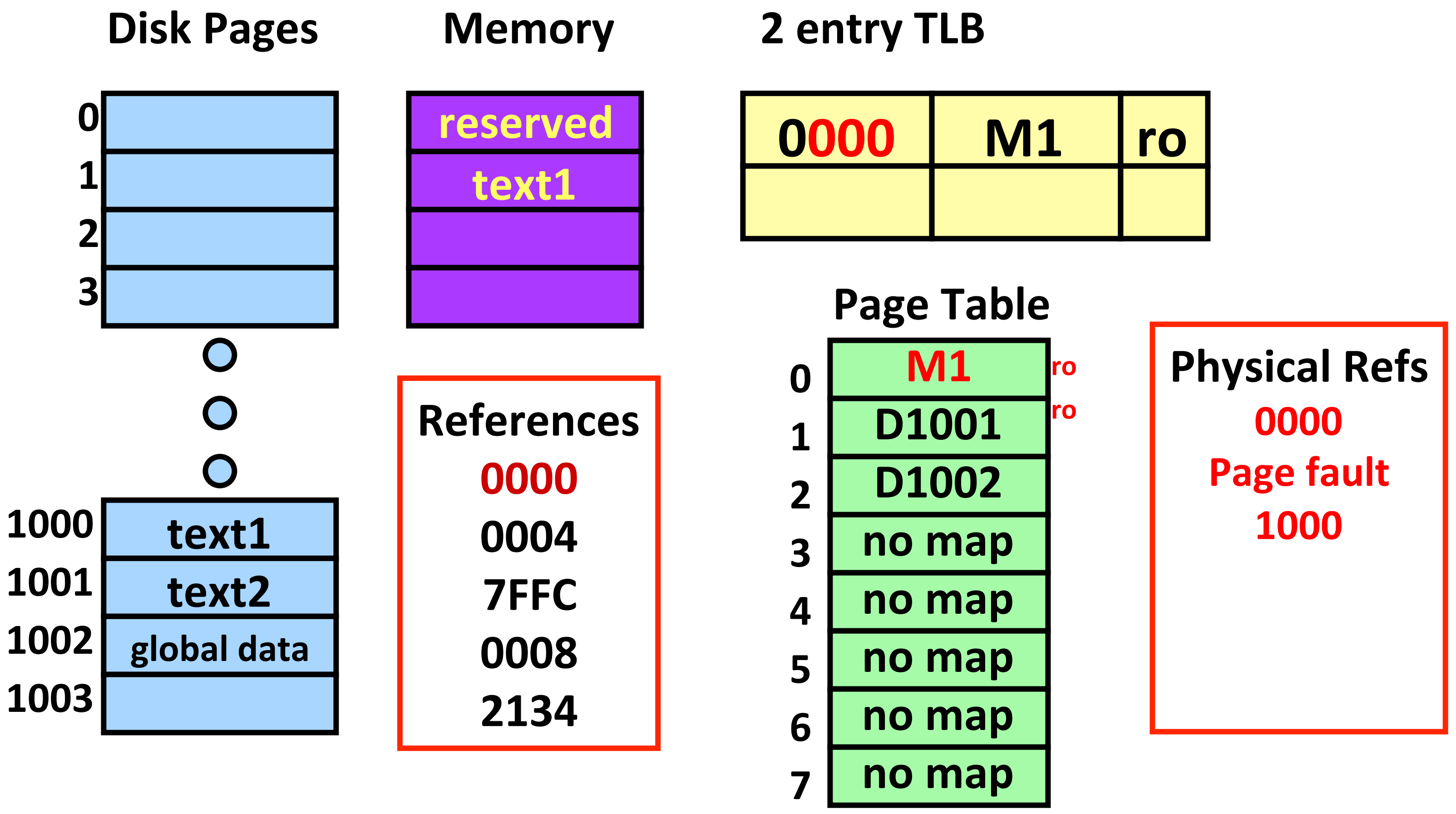
Fetching instruction 0000



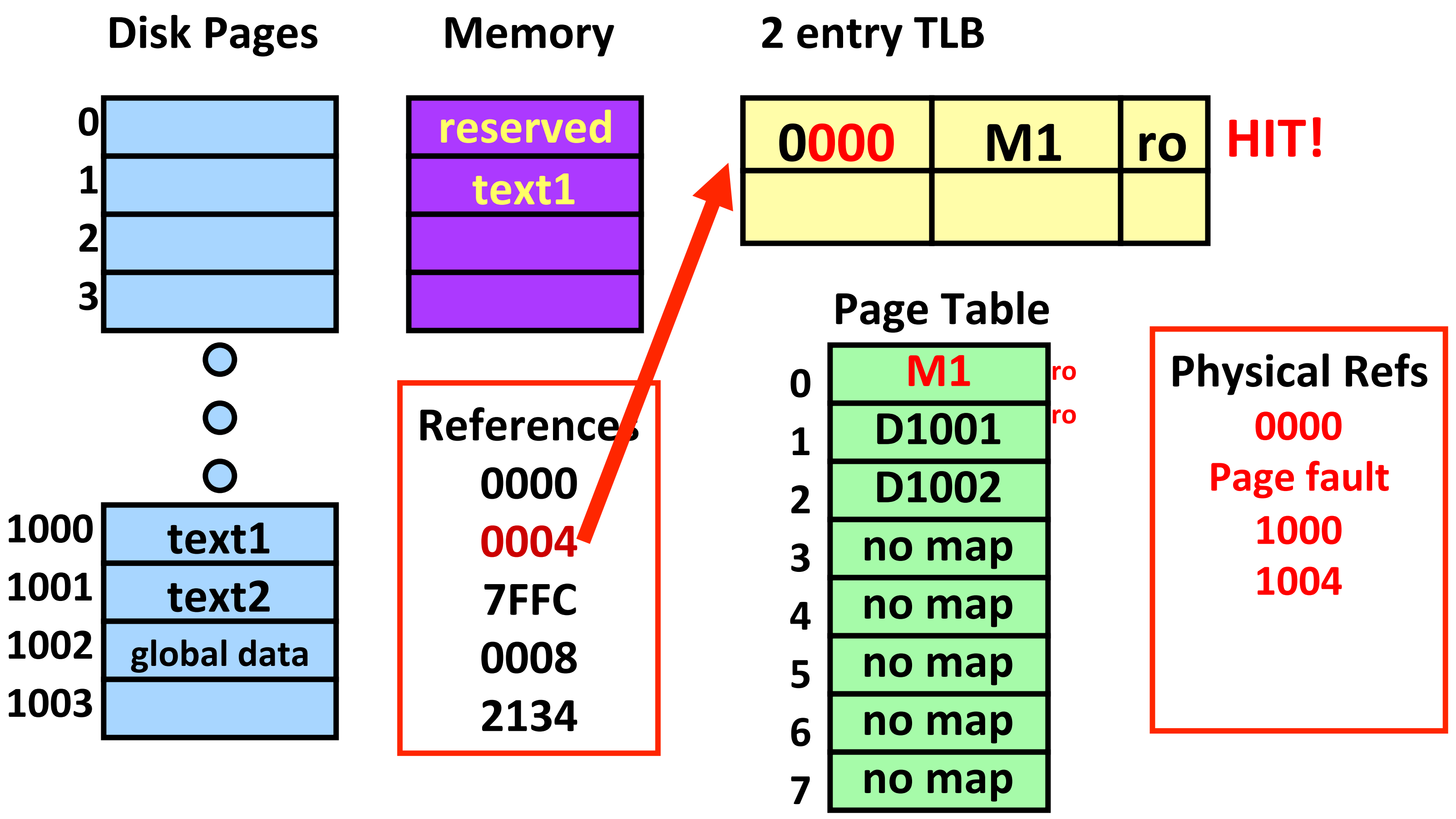
Fetching instruction 0000



Fetching instruction 0000



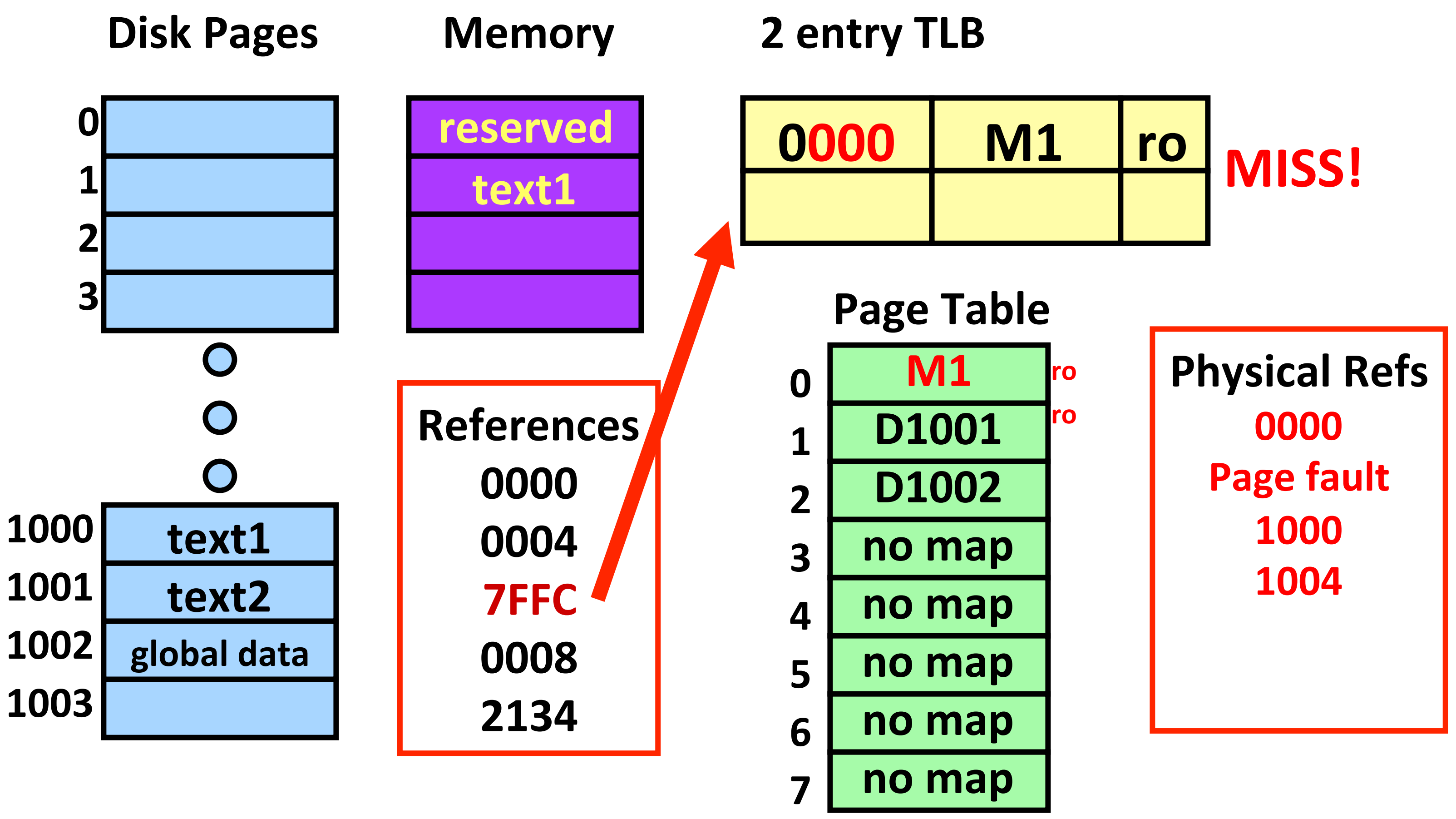
Fetching instruction 0004



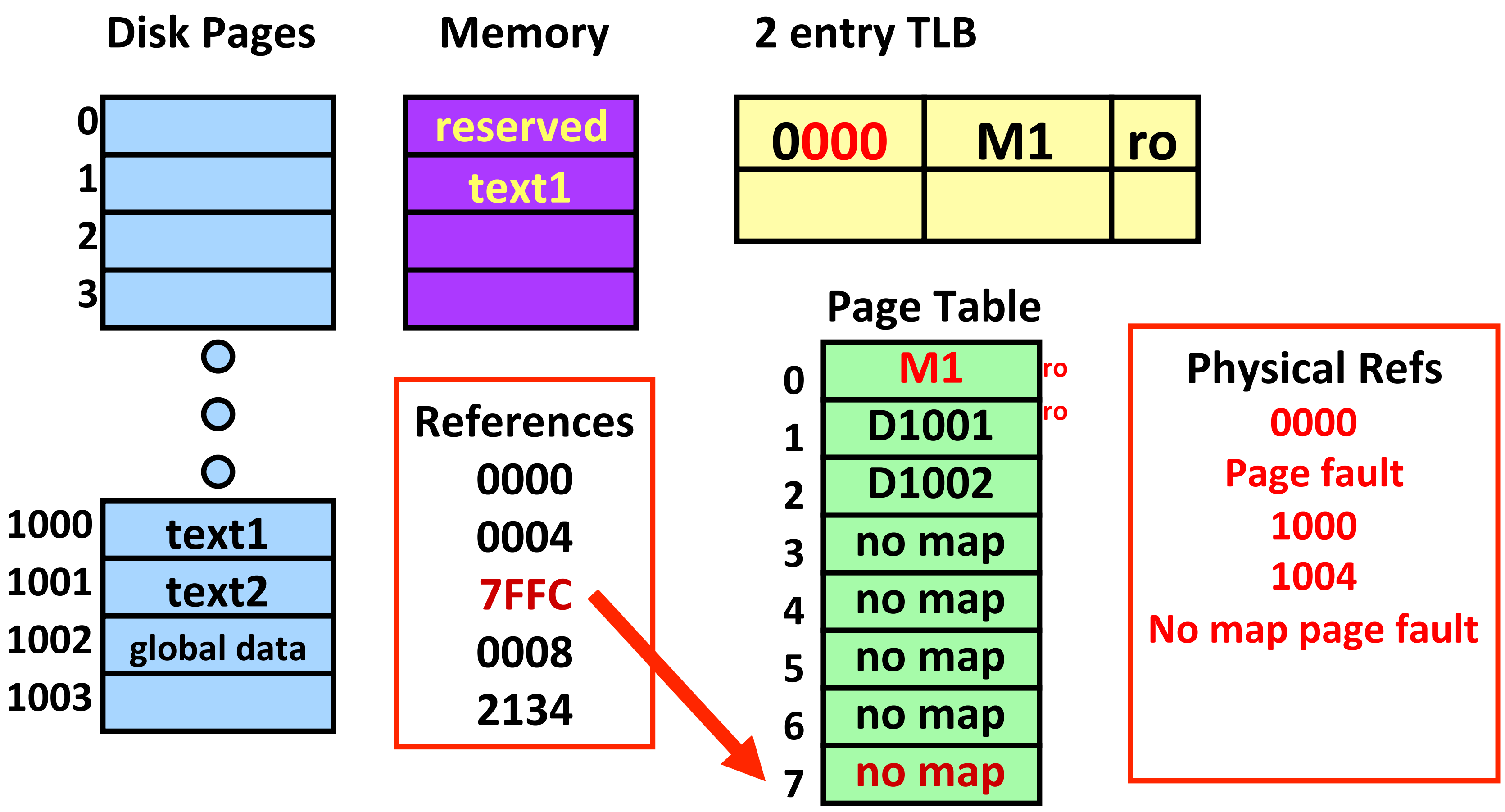
References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004

Reference 7FFC



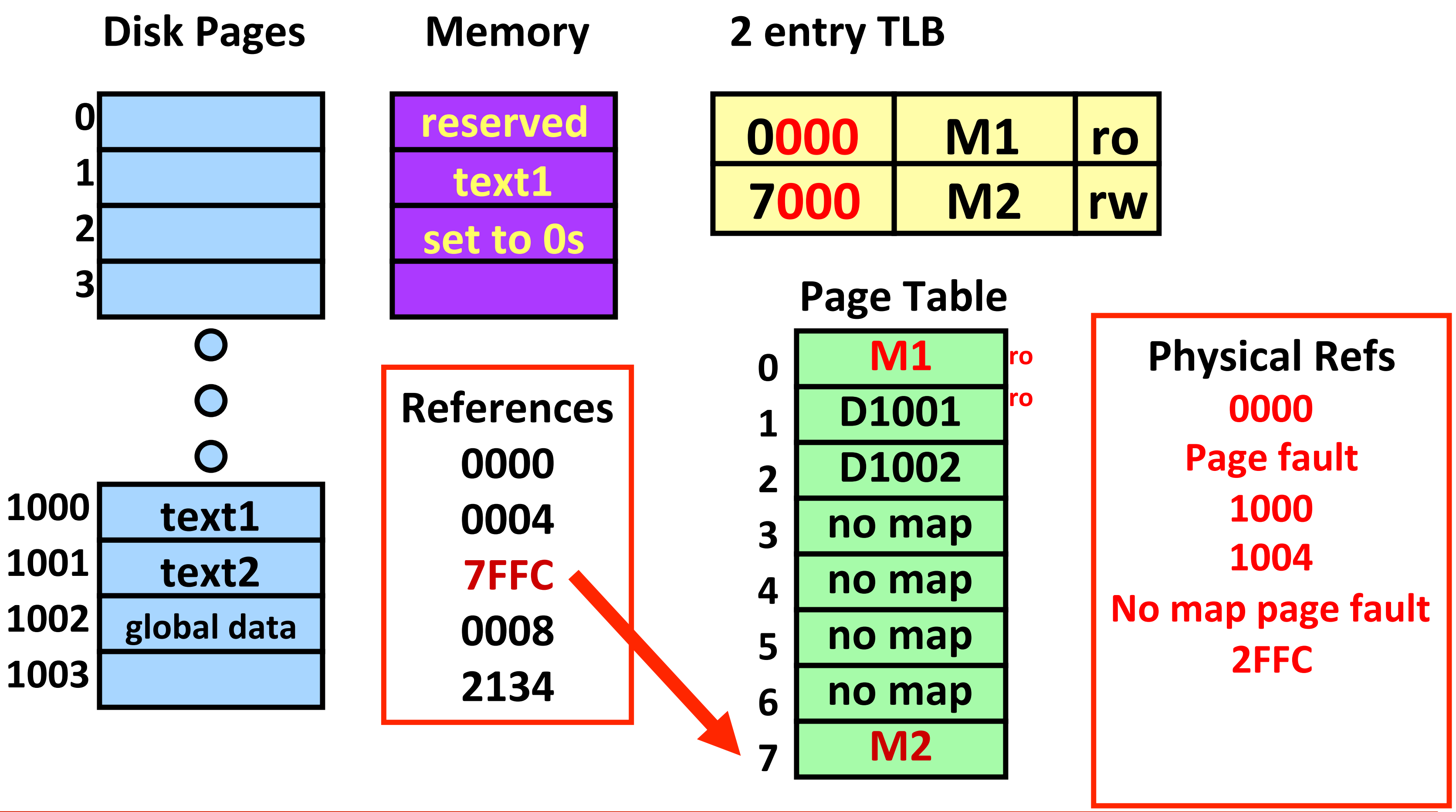
Reference 7FFC



References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004
No map page fault

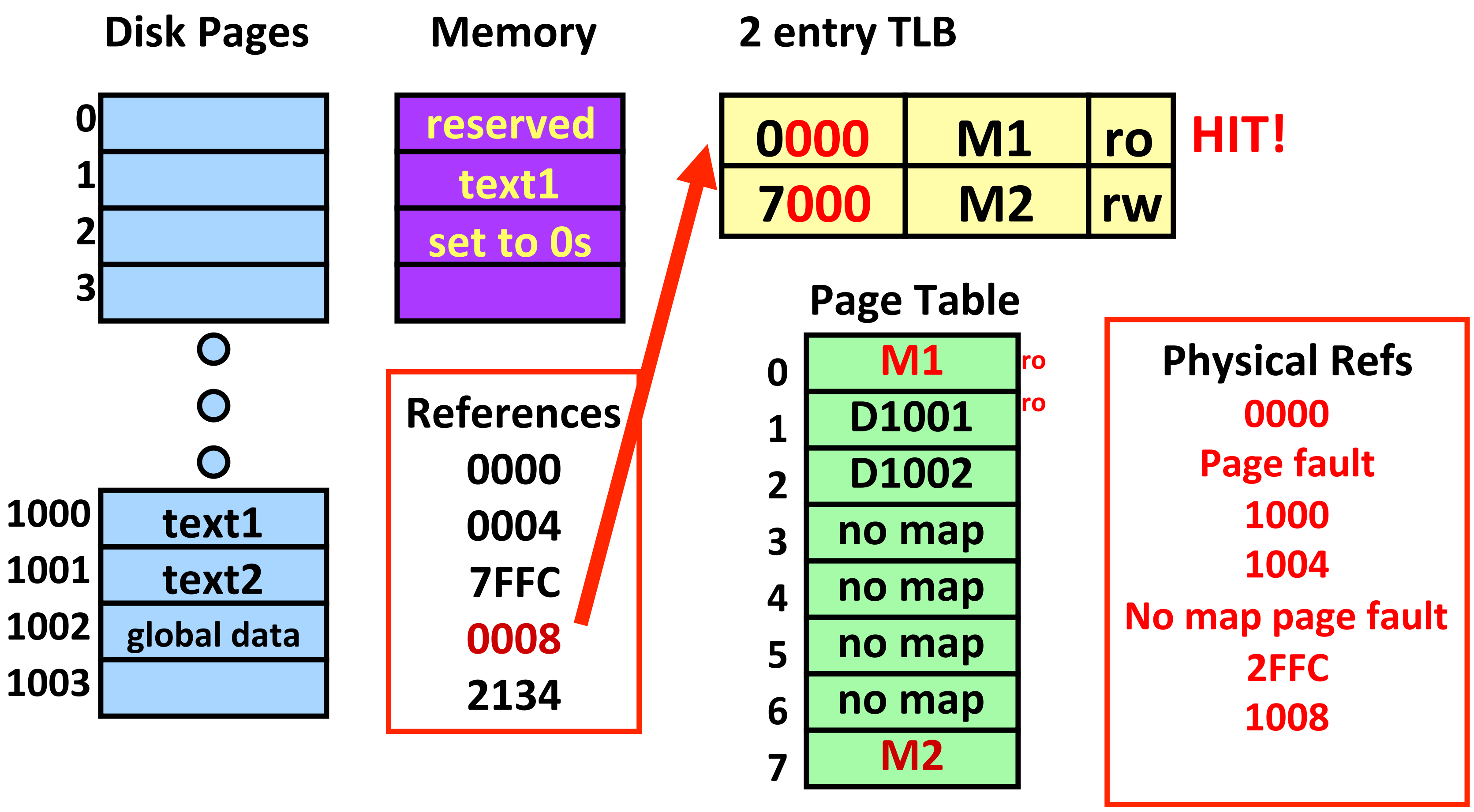
Reference 7FFC



References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004
No map page fault
2FFC

Fetching instruction 0008

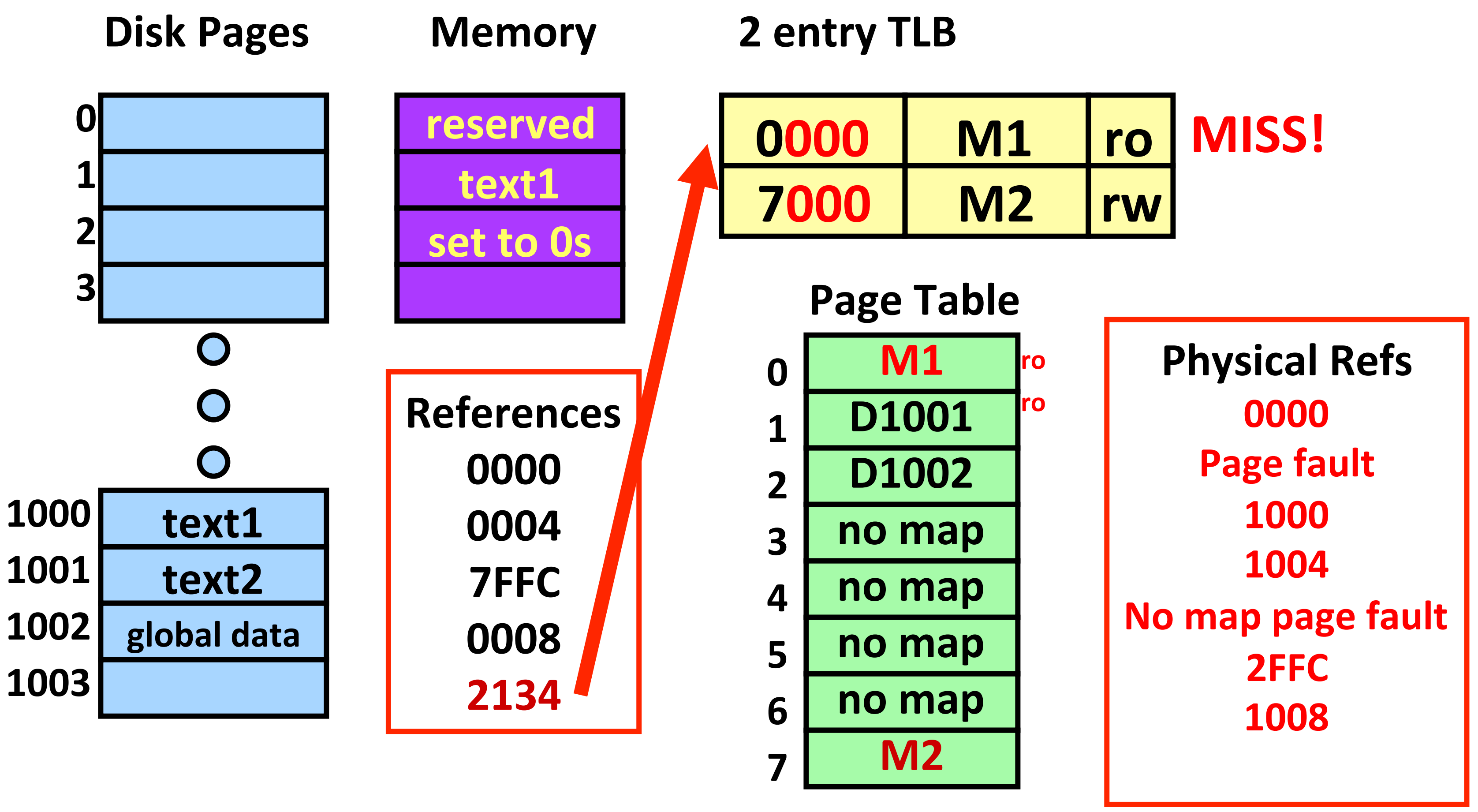


0000	M1	ro
7000	M2	rw

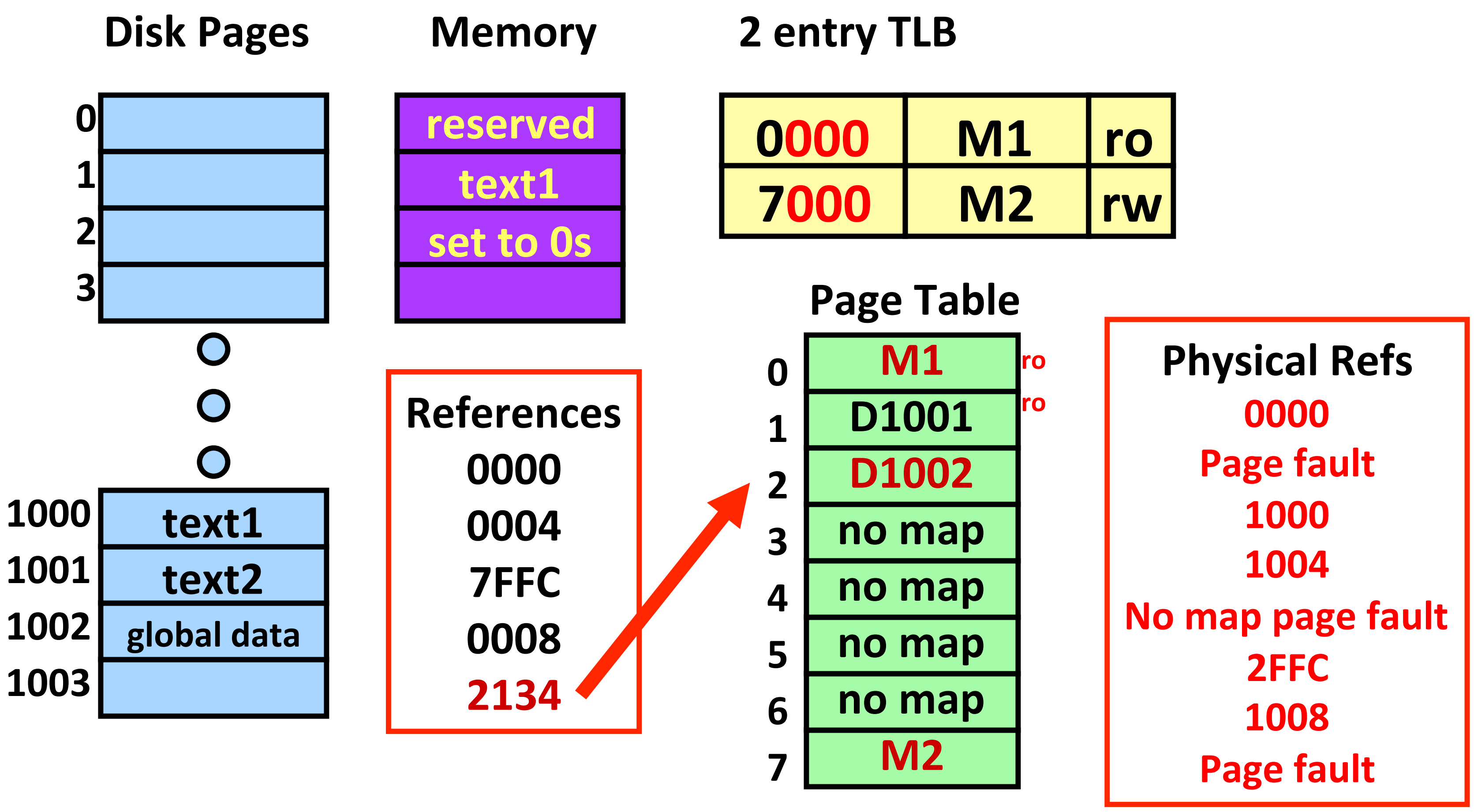
References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004
No map page fault
2FFC
1008

Reference 2134



Reference 2134



Disk Pages

0	
1	
2	
3	
1000	text1
1001	text2
1002	global data
1003	

Memory

reserved
text1
set to 0s

2 entry TLB

0000	M1	ro
7000	M2	rw

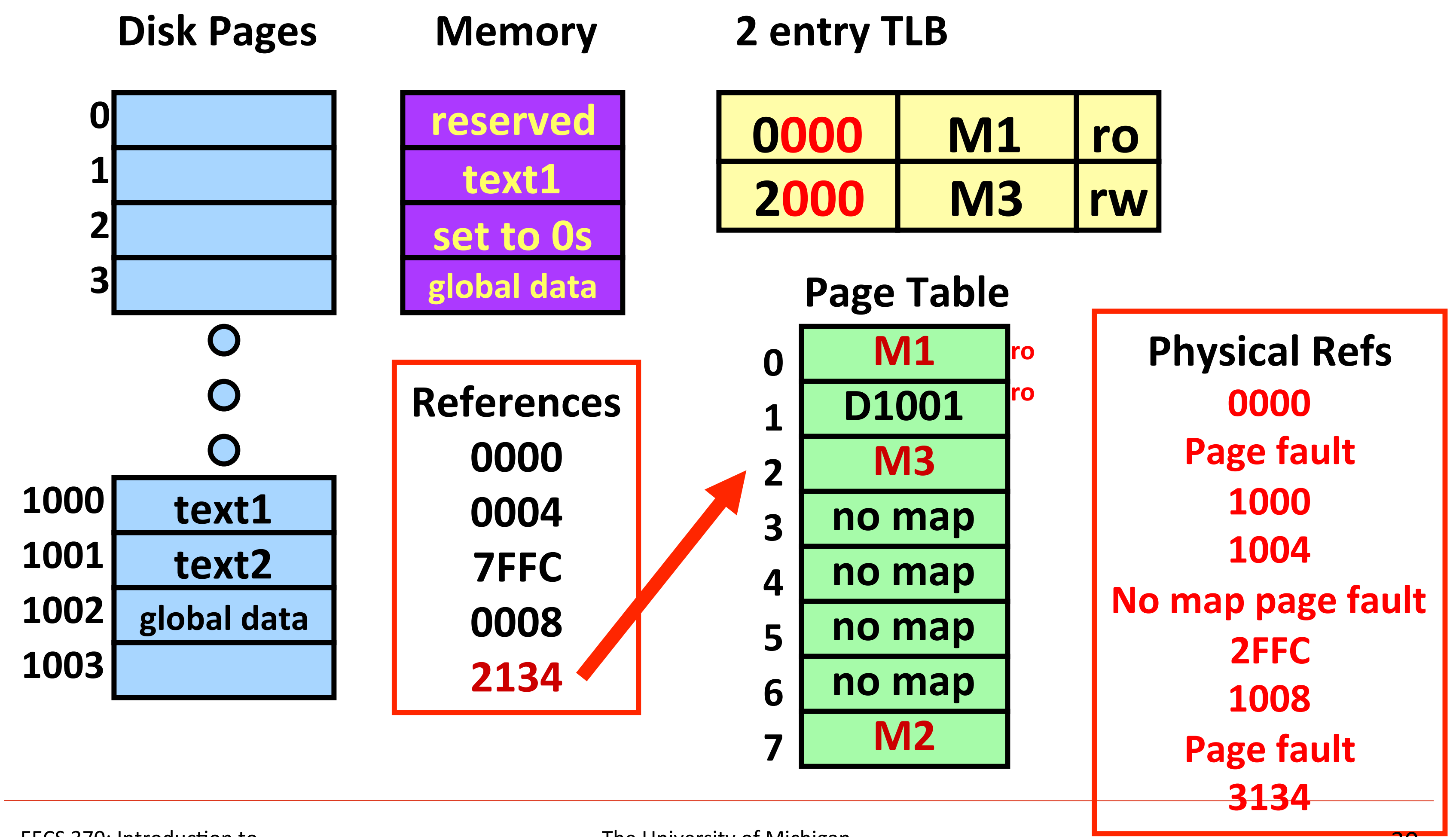
Page Table

0	M1	ro
1	D1001	ro
2	D1002	
3	no map	
4	no map	
5	no map	
6	no map	
7	M2	

References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004
No map page fault
2FFC
1008
Page fault

Reference 2134



0000	M1	ro
2000	M3	rw

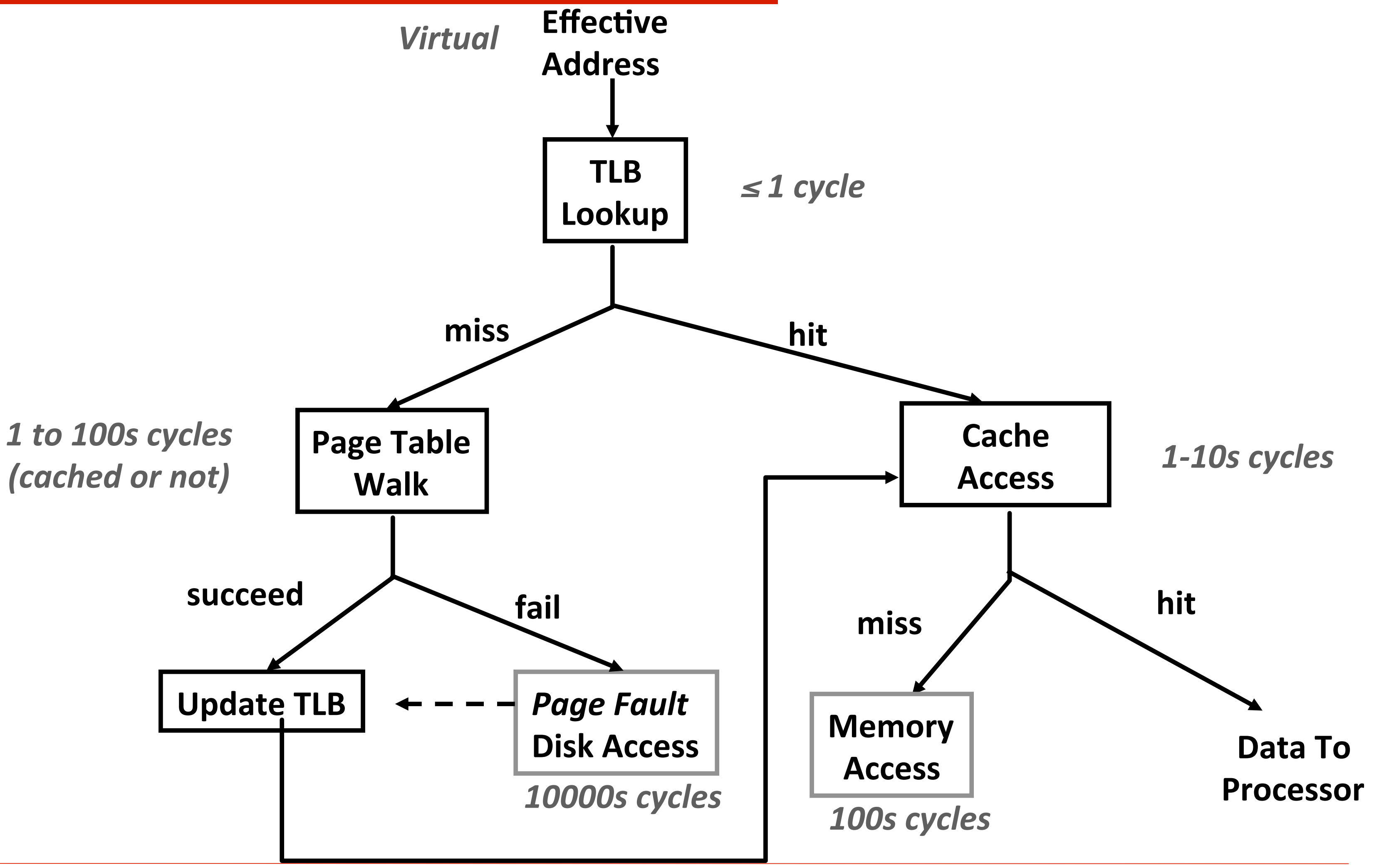
References
0000
0004
7FFC
0008
2134

Physical Refs
0000
Page fault
1000
1004
No map page fault
2FFC
1008
Page fault
3134

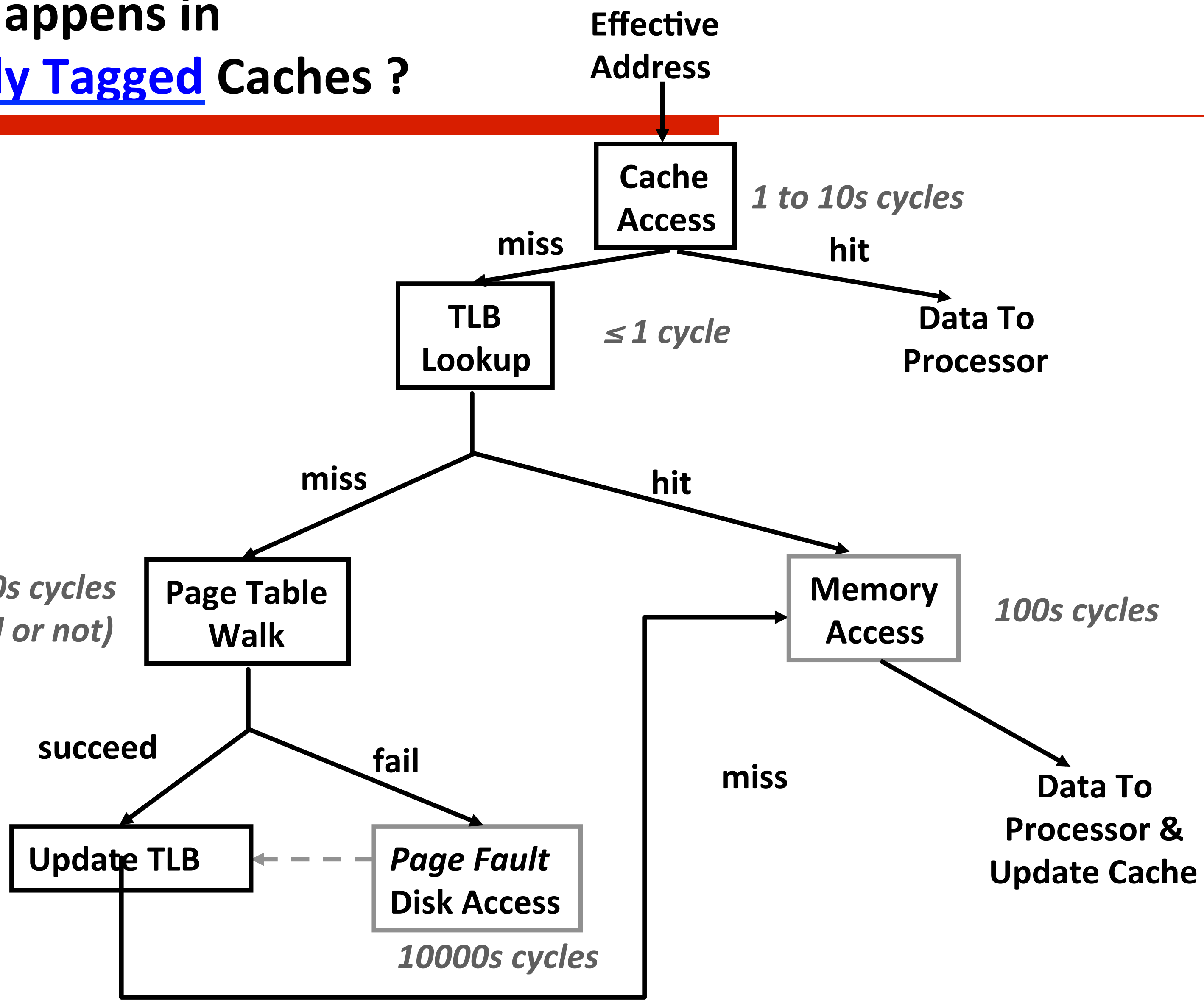
Multitasking with VM

- Flush the cache between each context switch.
- Use processID (a unique number for each processes given by the operating system) as part of the tag.

Physically tagged cache



What happens in Virtually Tagged Caches ?



Class problem – VM performance

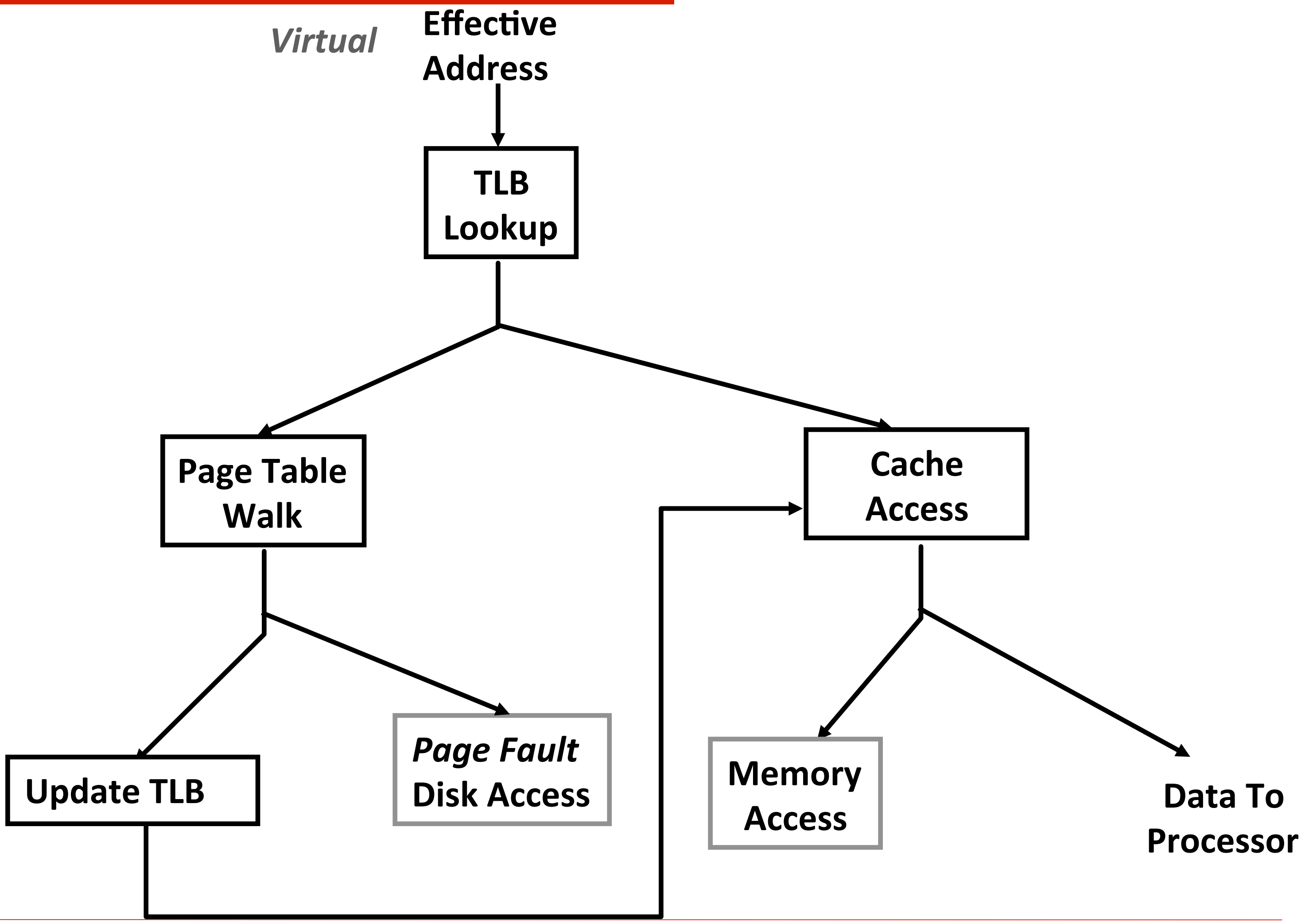
- Consider a system with unified data and instruction cache. The virtual memory system is a one-level page table system.
  - TLB hit rate: 99 %, TLB access time is 1 cycle
  - Cache hit rate: 95 %, cache access time is 1 cycle
  - Page fault rate: 0.01 % (that's 1 in ten thousand accesses)
  - Accesses to main memory require 30 cycles and hard drive require 100,000 cycles.

Notes:

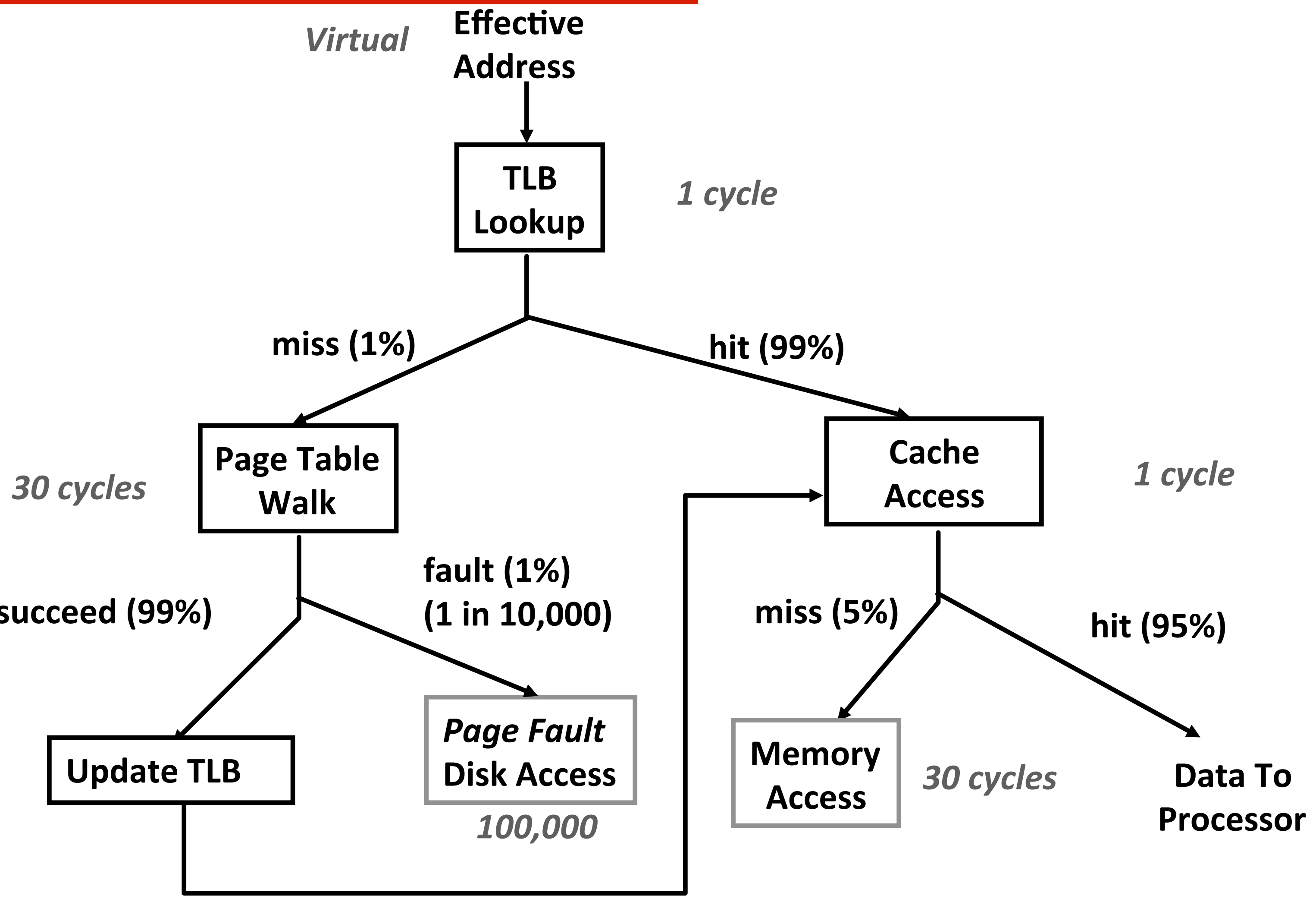
- The TLB access and cache access are sequential.
- TLB hits will not cause a page fault.
- The page table is uncacheable and always available in main memory.
- Upon retrieval from cache, main memory or hard drive, the data is sent immediately to the CPU, while other updates occur in parallel.

Compute the average latency of a memory access for this machine for virtual and physical tagged caches.

Class problem - Physically Addressed



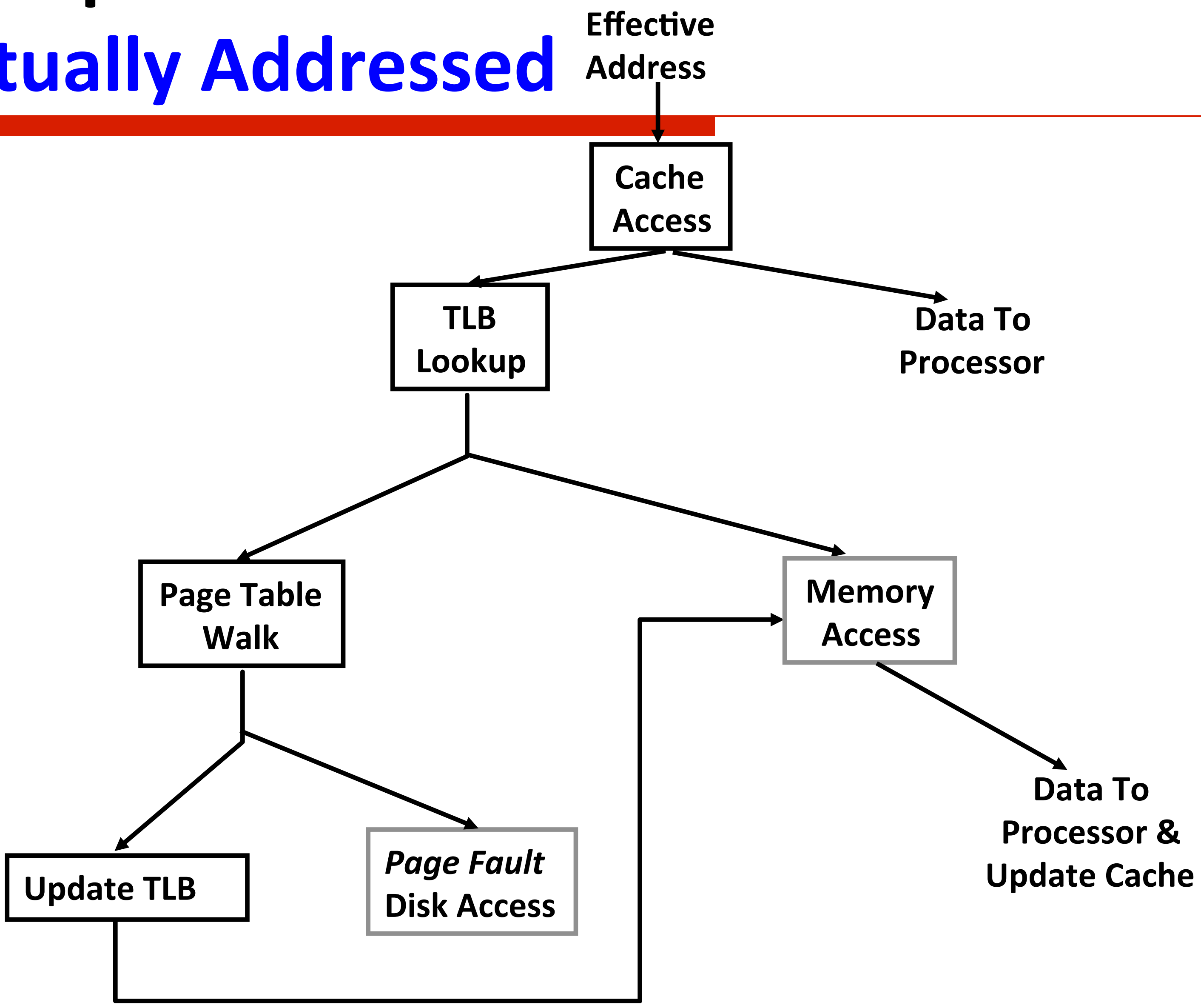
Class problem - Physically Addressed



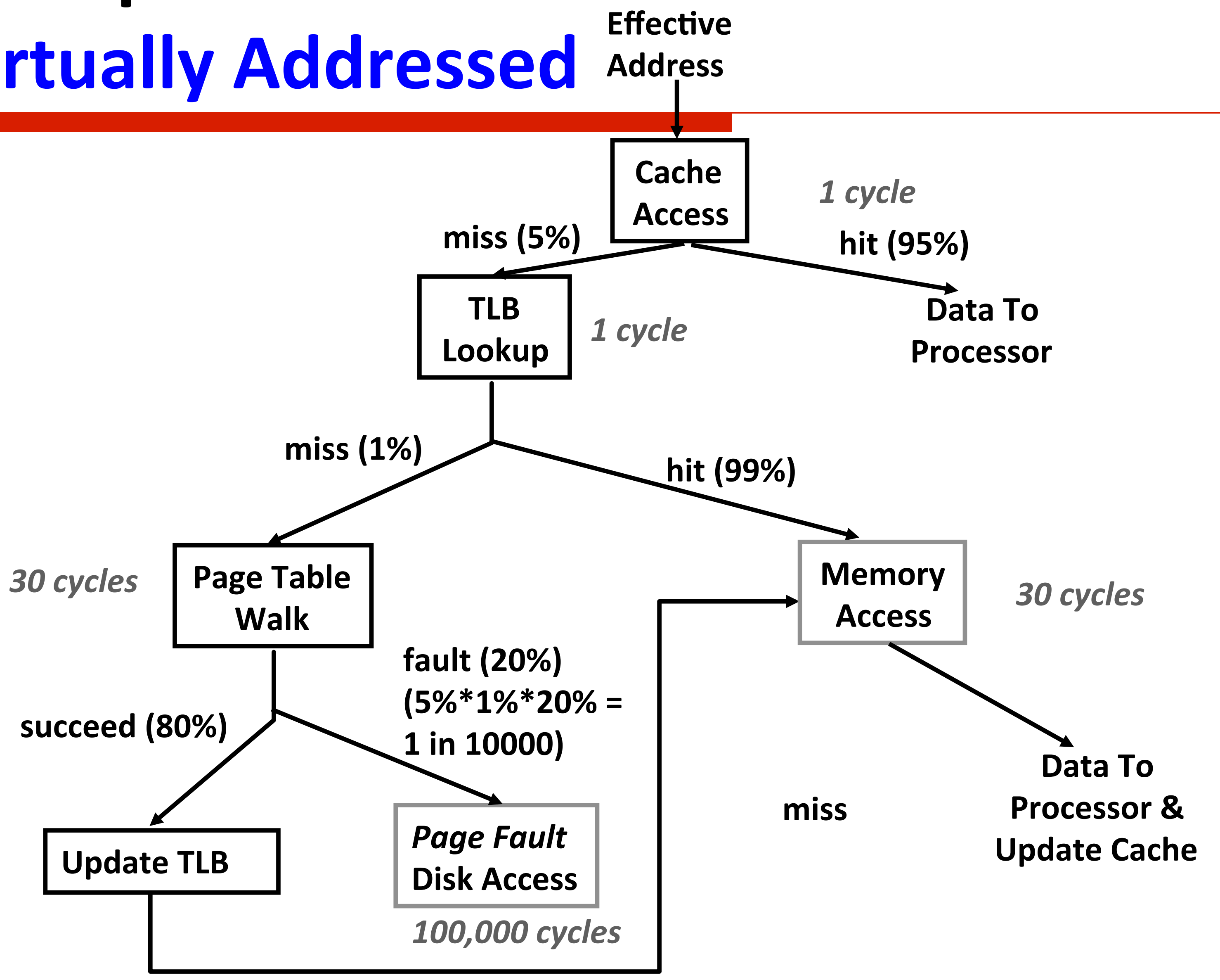
TLB + hit case + miss case

1 + 0.99 * (1+ .05 * 30) + .01 * [30 + 0.99 * (1 + .05 * 30) + .01 * 100,000]

Class problem - Virtually Addressed



Class problem - Virtually Addressed



cache access + miss case

1 + .05 * [1 + 0.99 * 30 + .01 (30 + 0.8 * 30 + .2 * 100,000)]