MODULE 7: Memory Systems

Lecture 7.3
Virtual Memory

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech

Lecture 7.3 Objectives

- Define virtual memory and describe the motivations for employing virtual memory
- Explain how paging works and what happens when there is a page fault
- Discuss what happens when both cache and virtual memory are employed in the same system
- Motivate the need for a Translation Look-aside Buffer and explain how it works
- Enumerate the tradeoffs between paging and segmentation

Virtual Memory

- Use of secondary storage (disk) to extend the apparent size of physical memory
- Cache memory enhances performance by providing faster memory access speed
- Virtual memory enhances performance by providing greater memory capacity at low cost
- Memory references issued by the CPU are translated from the logical address space to the physical address space
- Translation between physical address and virtual address is done in hardware for speed

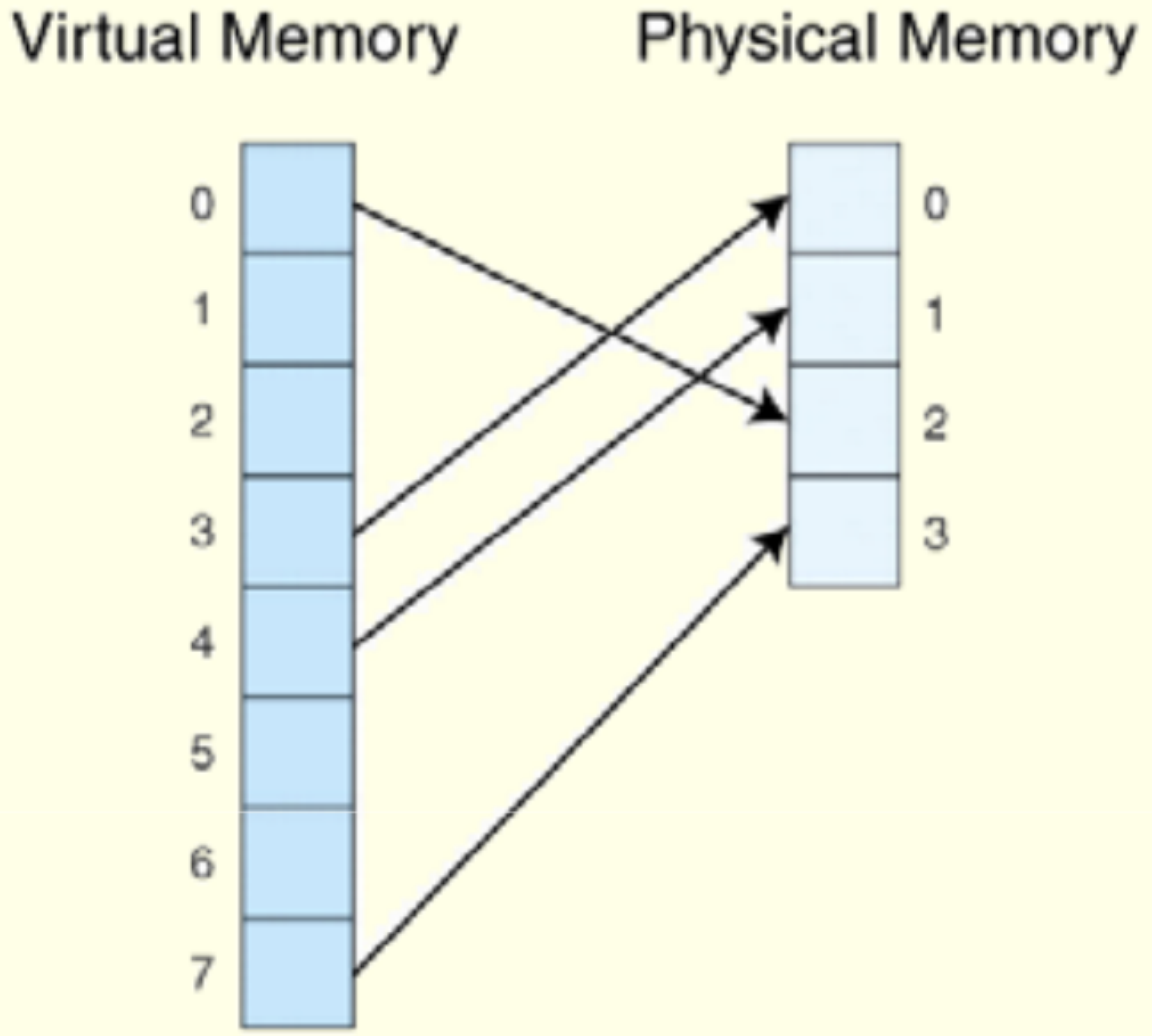
Paging

- Address space partitioned into equal sized blocks: pages
- Demand paging: a page is moved from disk onto main memory only when the processor accesses a word on that page
- Page faults occur when a logical address requires that a page be brought in from disk
- Memory fragmentation occurs when the paging process results in the creation of small, unusable clusters of memory addresses

Page Table

- Information concerning the location of each page, whether on disk or in memory, is maintained in a data structure called a page table
  - Page #s refer to virtual memory
  - Page frames refer to physical memory
  - Present bit: 0 = not in physical memory; 1 = in physical memory
- There is one page table for each active process

Page Table Example



Page Table

Page	Frame #	Valid Bit
0	2	1
1	-	0
2	-	0
3	0	1
4	1	1
5	-	0
6	-	0
7	3	1

Virtual and Physical Addresses

- When a process generates a virtual address, the operating system translates it into a physical memory address
- To accomplish this, the virtual address is divided into two fields
  - A page field determines the page location of the address
  - The offset field indicates the location of the address within the page
- The logical page number is translated into a physical page frame through a lookup in the page table

CHECK POINT

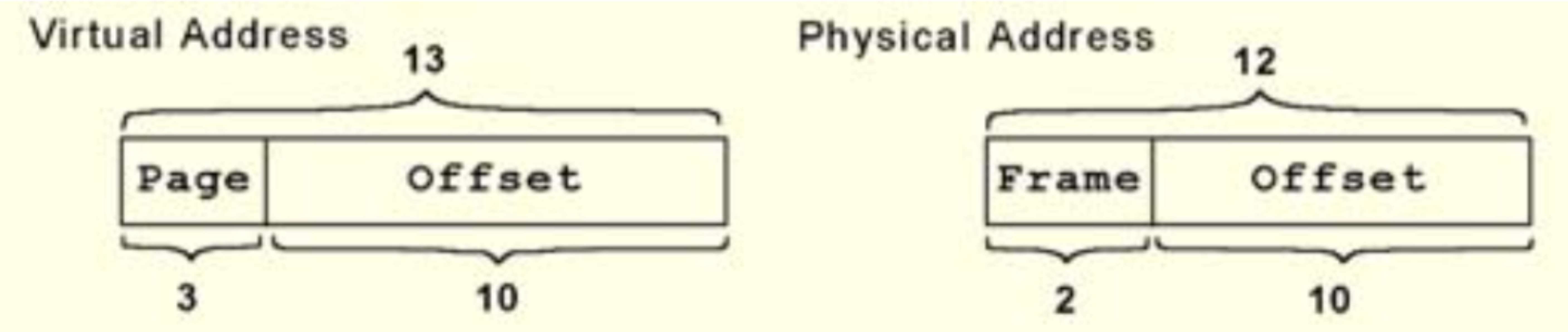
As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Define virtual memory and describe the motivations for employing virtual memory

If you have any difficulties, please review the lecture video before continuing.

Virtual Address Example

- Suppose a system has a virtual address space of 8KB and a physical address space of 4KB, and the system uses byte addressing and 1KB pages
  - We have $2^{13}/2^{10} = 2^3$ virtual pages
- A virtual address has 13 bits with 3 bits for the page field and 10 for the offset, because the page size is 1024
- A physical memory address requires 12 bits, the first two bits for the page frame and the trailing 10 bits the offset



Virtual Memory Example (1)

- Suppose we have the page table shown below
- What happens when CPU generates address $5459_{10} = 1010101010011_2$?

Page Table

Page	Frame	Valid Bit
0	–	0
1	3	1
2	0	1
3	–	0
4	–	0
5	1	1
6	2	1
7	–	0

Addresses

Page	Addresses
0	0 - 1023
1	1024 - 2047
2	2048 - 3071
3	3072 - 4095
4	4096 - 5119
5	5120 - 6143
6	6144 - 7167
7	7168 - 8191

Virtual Memory Example (2)

- The address 1010101010011_2 is converted to physical address 010101010011_2 because the page field 101 is replaced by frame number 01 through a lookup in the page table

Page Table

Page	Frame	Valid Bit	Addresses
0	-	0	0 - 1023
1	3	1	1024 - 2047
2	0	1	2048 - 3071
3	-	0	3072 - 4095
4	-	0	4096 - 5119
5	1	1	5120 - 6143
6	2	1	6144 - 7167
7	-	0	7168 - 8191

Page Faults

- When a referenced virtual location is currently not in physical memory, a page fault occurs:

1. Identify a page frame in physical memory to be overwritten; write contents of this frame to secondary storage (if changes have been made to it)
2. Locate desired virtual page in secondary memory, write it to physical memory
3. Update page table to map new section of the virtual memory onto physical memory
4. Continue execution

CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Explain how paging works and what happens when there is a page fault

If you have any difficulties, please review the lecture video before continuing.

EAT Revisited

- Virtual memory is also a factor in the calculation of EAT
  - We also have to consider page table access time
- Suppose a main memory access takes 200ns, the page fault rate is 1%, and it takes 10ms to load a page from disk

EAT = 0.99 (200 ns + 200 ns) + 0.01 x 10 ms = 100.396 μs

time to reference the page table

time to reference the actual data

Translation Look-aside Buffer

- With virtual memory, need 2 references to access a value in memory: reference to the page table to find physical page frame, and reference to actual data value
- Maintain some small amount of memory inside the CPU that stores most recent translations between virtual and physical addresses – the TLB
- TLB is searched first for virtual addresses
- If a TLB miss occurs, the virtual address is looked up in the page table in main memory and the TLB is updated

Cache and Virtual Memory

- Suppose a computer uses cache AND virtual memory
- When a memory reference is made:
    1. Satisfy the reference if the referenced word is in the cache
    2. If not, read the block containing the referenced word into the cache from main memory
    3. If not in main memory, bring the page containing the word into main memory from disk, load appropriate block into the cache
- Note: pages are large as compared to cache memory blocks

Segmentation

- Instead of dividing memory into equal-sized pages, virtual address space is divided into variable-length segments, often under the control of the programmer
    - Segments are brought in from secondary memory as needed
- Segments can be specified as “read only,” “execute only,” etc. for protection
- Programs specify segment number and an address within the segment; OS translates segmented address into physical address
- Not as common as paged virtual memory

Paging and Segmentation

- Large page tables are cumbersome and slow, but with its uniform memory mapping, page operations are fast
- Segmentation allows fast access to the segment table, but segment loading is labor-intensive
- Paging and segmentation can be combined to take advantage of the best features of both by assigning fixed- size pages within variable-sized segments
  - Each segment has a page table
  - A memory address will have three fields: one for the segment, another for the page, and a third for the offset

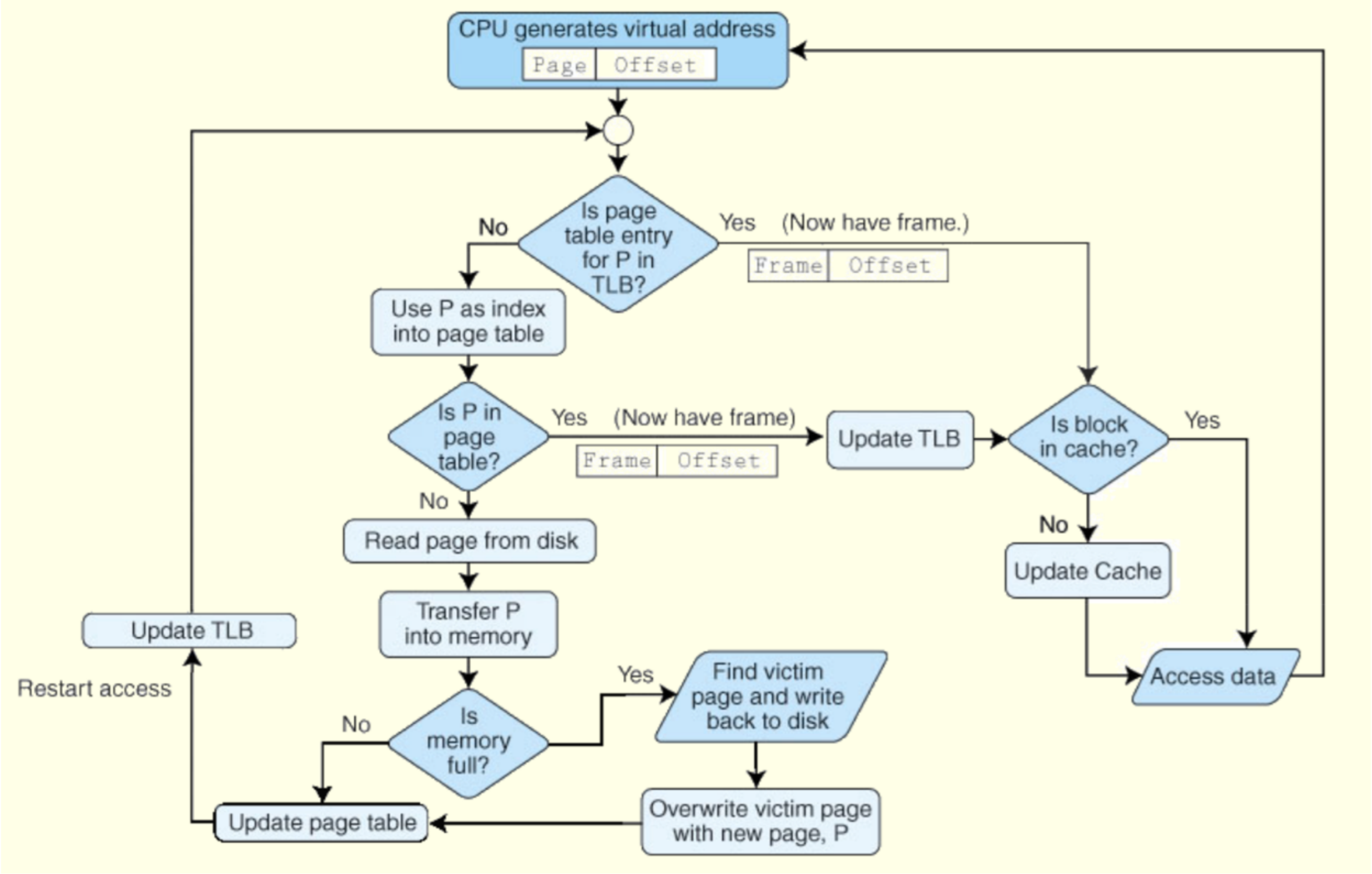
CHECK POINT

As a checkpoint of your understanding, please pause the video and make sure you can do the following:

- Discuss what happens when both cache and virtual memory are employed in the same system
- Motivate the need for a Translation Look-aside Buffer and explain how it works
- Enumerate the tradeoffs between paging and segmentation

If you have any difficulties, please review the lecture video before continuing.

Summary


CPU generates virtual address
Page
Offset
Is page table entry for P in TLB?
No
Yes
(Now have frame.)
Frame
Offset
Use P as index into page table
Is P in page table?
Yes
(Now have frame)
Frame
Offset
Update TLB
Is block in cache?
Yes
No
Update Cache
Access data
No
Read page from disk
Transfer P into memory
Update TLB
Restart access
Is memory full?
No
Yes
Find victim page and write back to disk
Overwrite victim page with new page, P
Update page table

MODULE 7: Memory Systems

Lecture 7.3
Virtual Memory

Prepared By:
- Scott F. Midkiff, PhD
- Luiz A. DaSilva, PhD
- Kendall E. Giles, PhD

Electrical and Computer Engineering
Virginia Tech