COL788: Advanced Topics in Embedded Computing

Lecture 13 – Memory System

Vireshwar Kumar
CSE@IITD

September 07, 2022

Semester I
2022-2023

Memory

- Many programs run at the same time
- The CPU of course runs one program at a time
  - Switches between programs periodically
  - Program states are stored in the memory

Memory Types

- Tradeoffs: Area, Power, and Latency
  - Increase Area → Reduce latency, increase power
  - Reduce latency → increase area, increase power
  - Reduce power → reduce area, increase latency

Cell Type	Area	Typical Latency
Master Slave D flip flop	0.8 μm^2	Fraction of a cycle
SRAM cell in an array	0.08 μm^2	1-5 cycles
DRAM cell in an array	0.005 μm^2	50-200 cycles

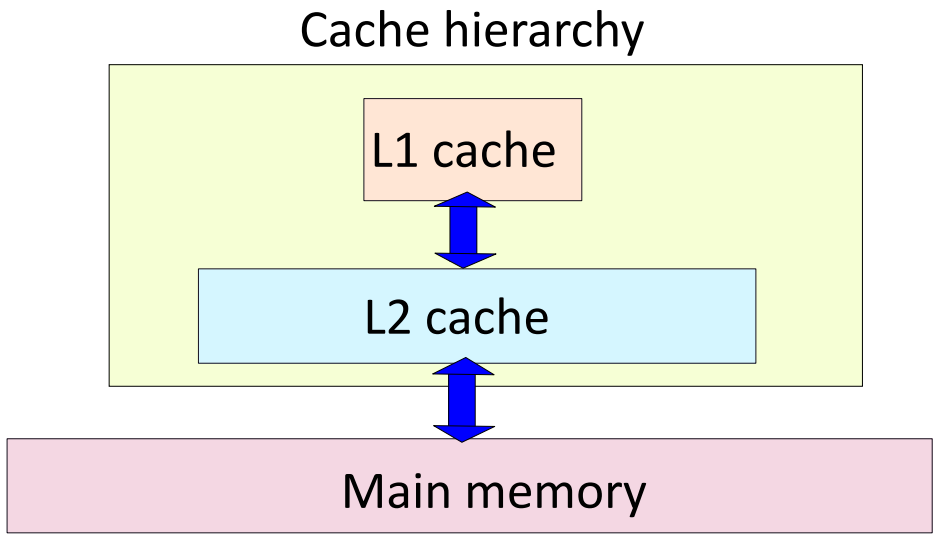
Temporal and Spatial Locality

- Temporal Locality
  - If a resource is accessed at some point of time, then most likely it will be accessed again in a short period of time.

- Spatial Locality
  - If a resource is accessed at some point of time, then most likely similar resources will be accessed again in the near future.

Exploiting Temporal Locality

- The L1 cache is a small memory (8-64 KB) composed of SRAM cells
- The L2 cache is larger and slower (128 KB – 4 MB) (SRAM cells)
- The main memory is even larger (1 – 64 GB) (DRAM cells)



Cache Hit and Miss

- Access Protocol
  - First access the L1 cache. If the memory location is present, we have a cache hit.
    - Perform the access (read/write)
  - Otherwise, we have a cache miss.
    - Fetch the value from the lower levels of the memory system, and populate the cache.
    - Follow this protocol recursively

- Typical Hit Rates, Latencies
  - L1 : 95 %, 1 cycle
  - L2 : 60 %, 10 cycles
  - Main Memory : 100 %, 300 cycles

Exploiting Spatial Locality

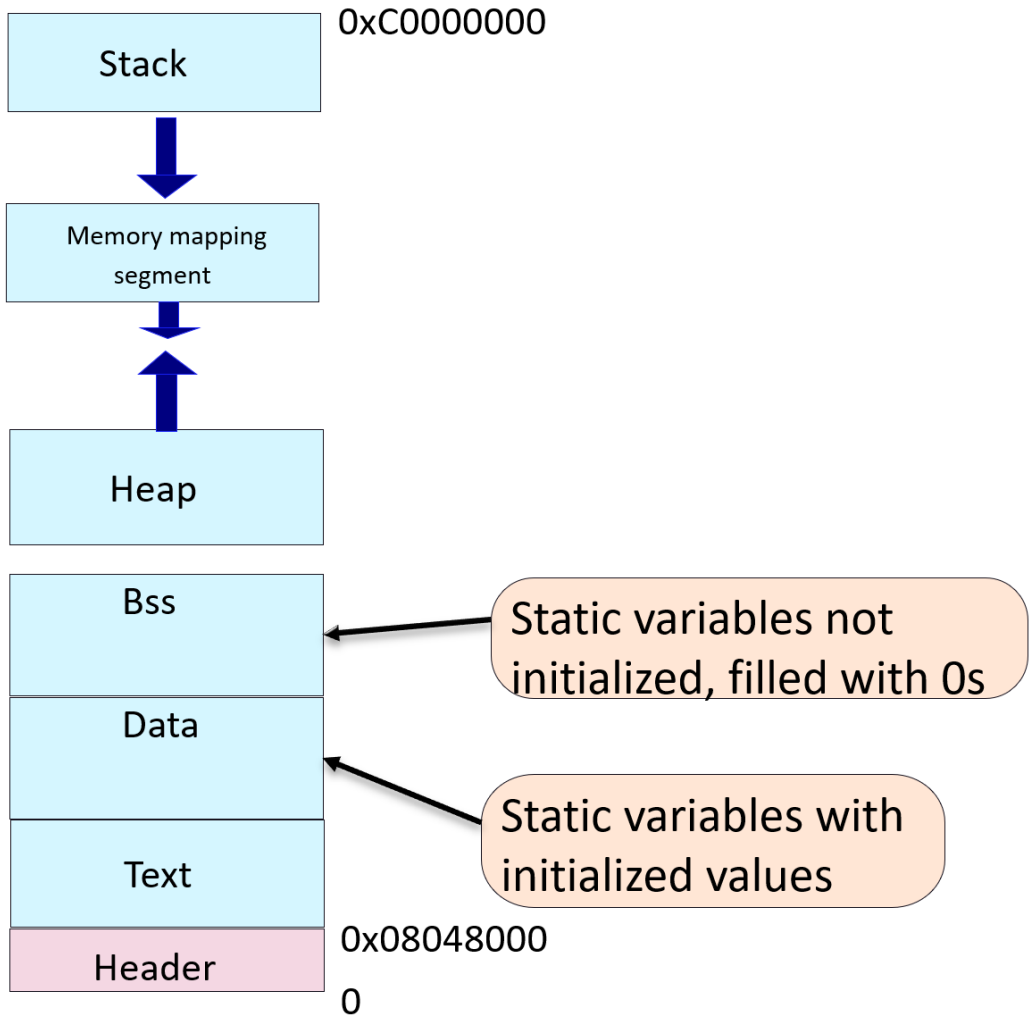
- Group memory addresses into sets of n bytes
- Each group is known as a cache line or cache block
- A cache block is typically 32, 64, or 128 bytes

Physical and Virtual Memory

- Physical Memory
  - Refers to the actual set of physical memory locations contained in the main memory, and the caches.

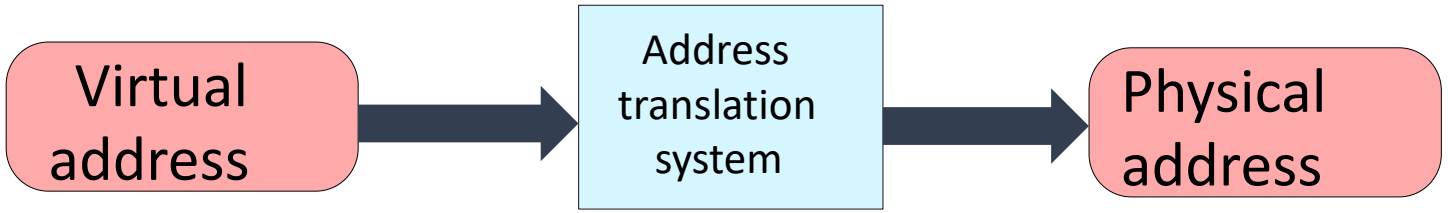
- Virtual Memory
  - The memory space assumed by a program.
  - Contiguous, without limits.

Virtual Memory Map Example

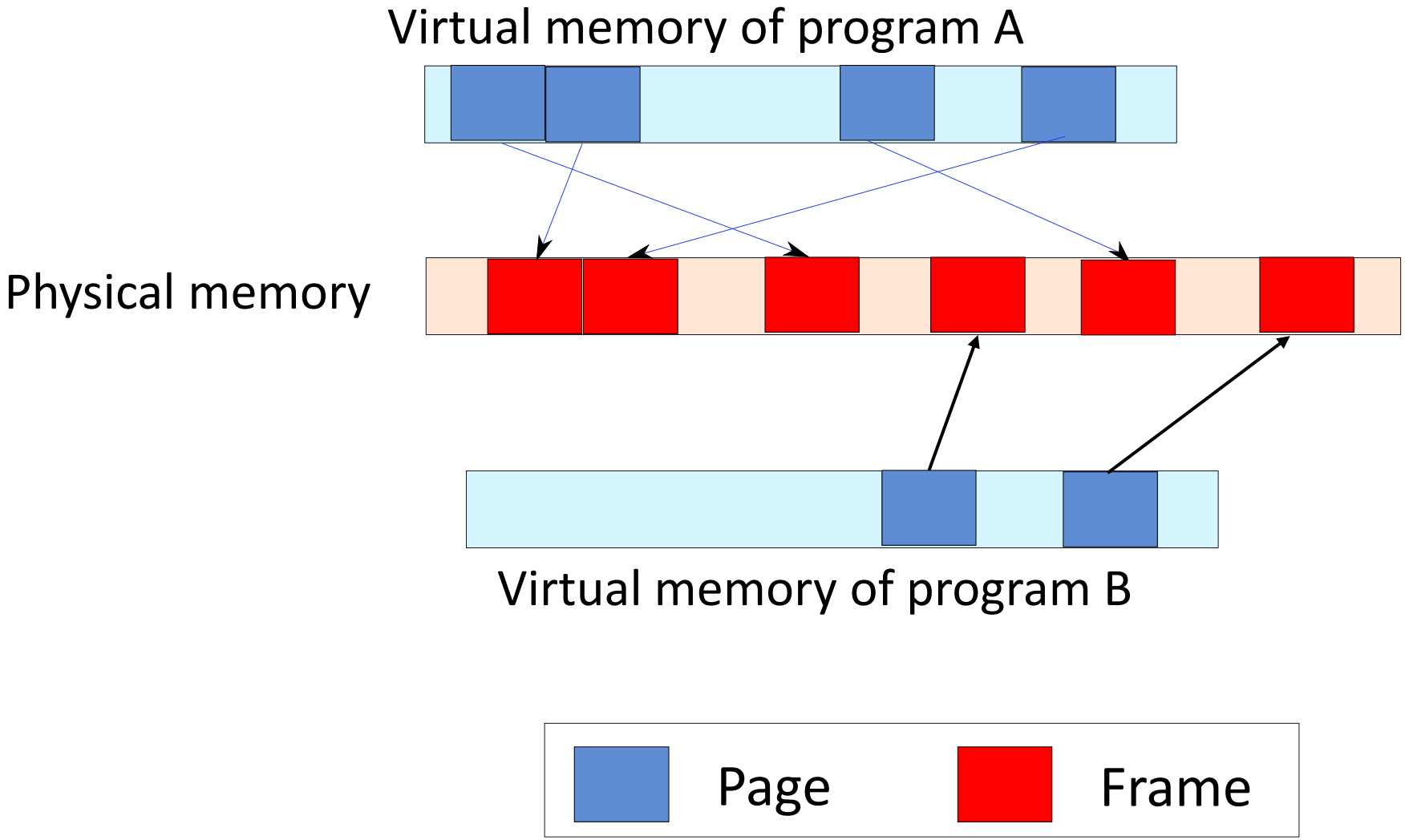


Address Translation

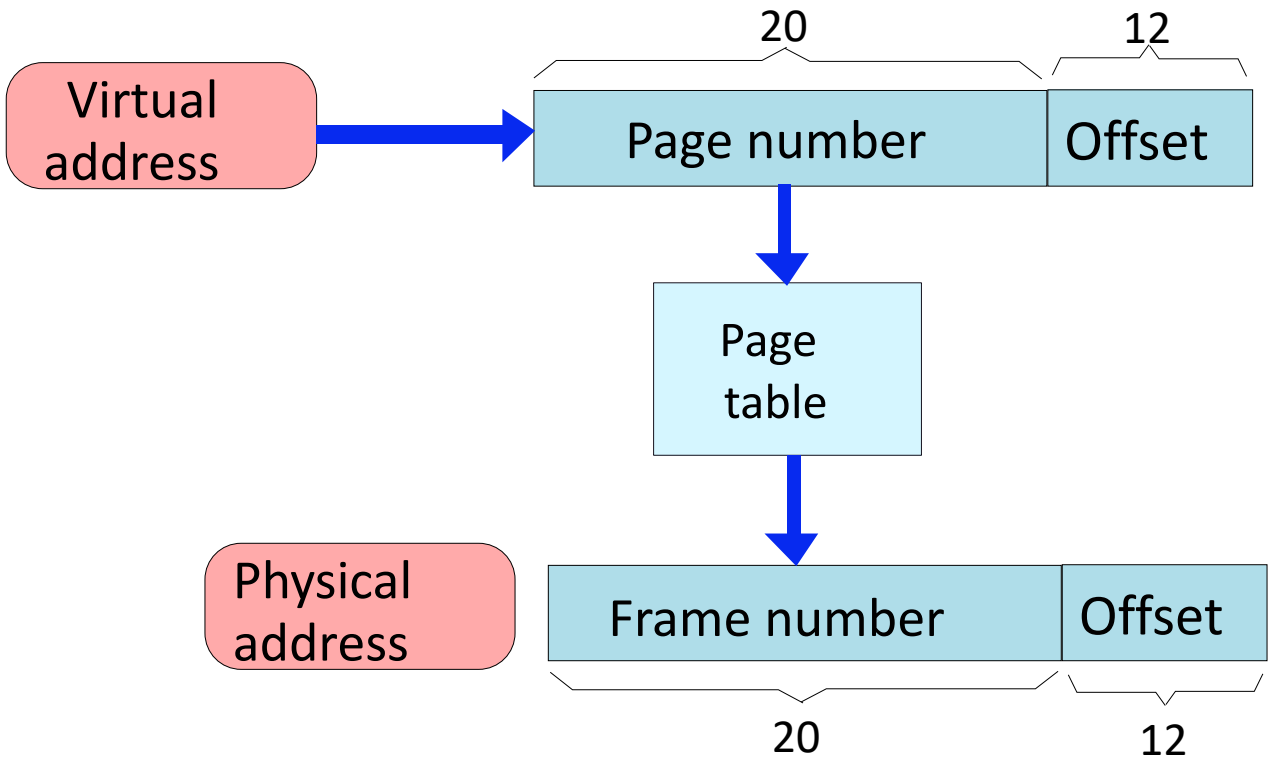
- Divide the virtual address space into chunks of 4 kB → page
- Divide the physical address space into chunks of 4 kB → frame
- Map pages to frames



Mapping Example

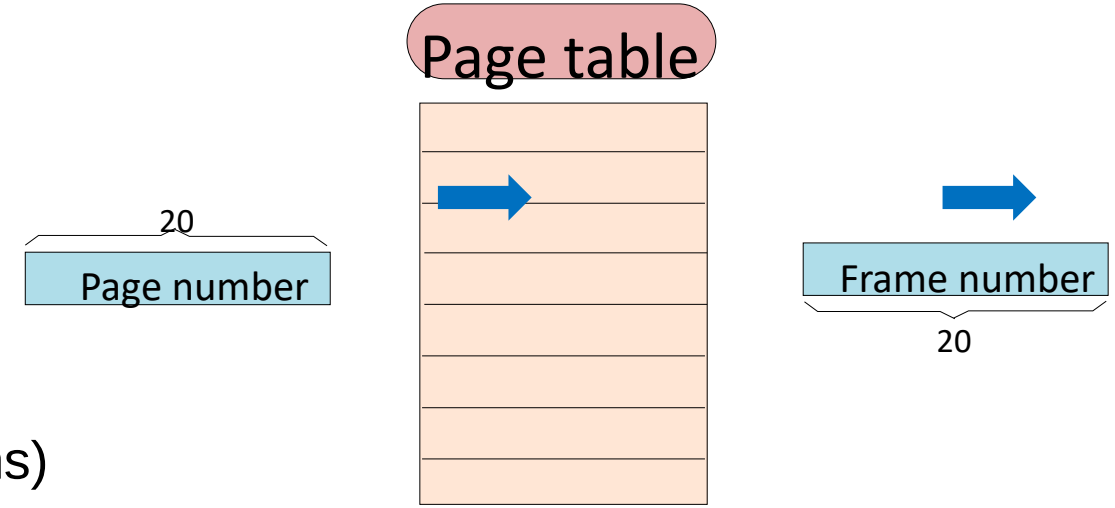


Page Translation Example



Single Level Page Table

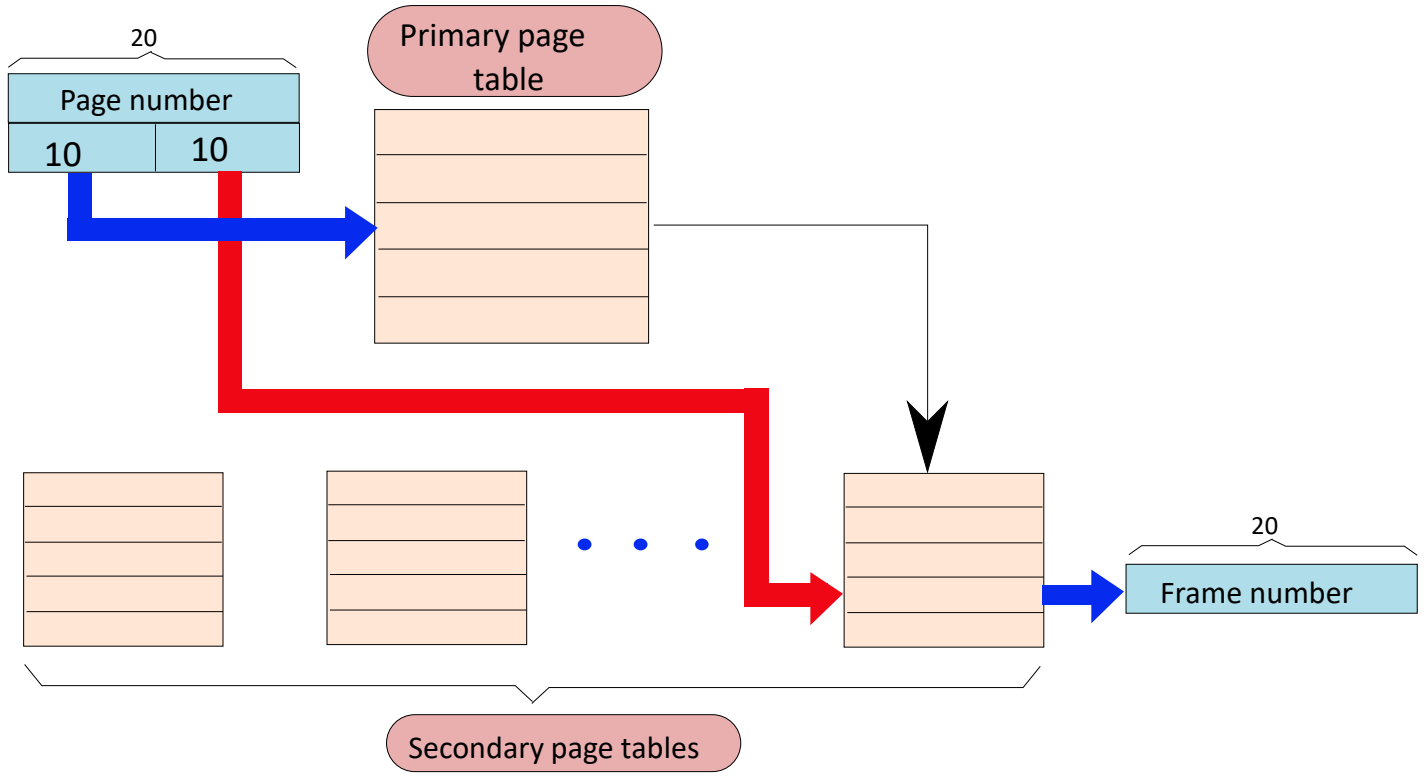
- Size of the single level page table
  - Size of an entry (20 bits = 2.5 bytes) *
  - Number of entries (220 = 1 million)
  - Total → 2.5 MB



- For 200 processes (running instances of programs)
  - We spend 500 MB in saving page tables (not acceptable)

- Implication
  - Most of the virtual address space is empty
  - Most programs do not require that much of memory

Two Level Page Table

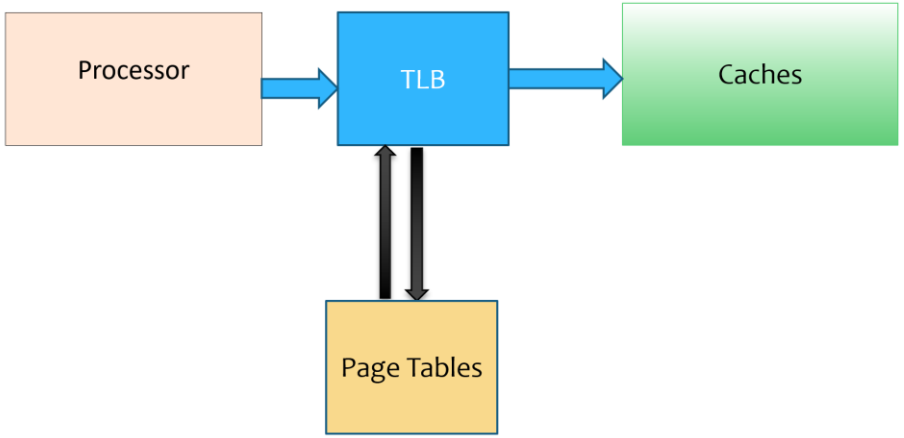


Advantages of Two Level Page Table

- We have a two level set of page tables
  - Primary and secondary page tables
- Not all the entries of the primary page table point to valid secondary page tables
- Each secondary page table → 1024 * 2.5 B = 2.5 KB
  - Maps 4MB of virtual memory
- Implications
  - Allocate only those many secondary page tables as required.
  - We do not need many secondary page tables due to spatial locality in programs
  - Example: If a program uses 100 MB of virtual memory and needs 25 secondary page tables, we need a total of 2.5KB * 25 = 62.5 KB of space for saving secondary page tables (minimal).

Translation Lookaside Buffer

- TLB (Translation Lookaside Buffer)
  - A fully associative cache
  - Each entry contains a page → frame (mapping)
  - Typically contains 64 entries
  - Very few accesses go to the page table



- Accesses that go to the page table
  - If there is no mapping, we have a page fault
  - On a page fault, create a mapping, and allocate an empty frame in memory. Update the list of empty frames.

What’s Next?

- Lecture 14
  - September 08, Thursday, 12 pm – 1 pm