Virtual Memory

Computer Organization

Monday, 30 September 2024

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

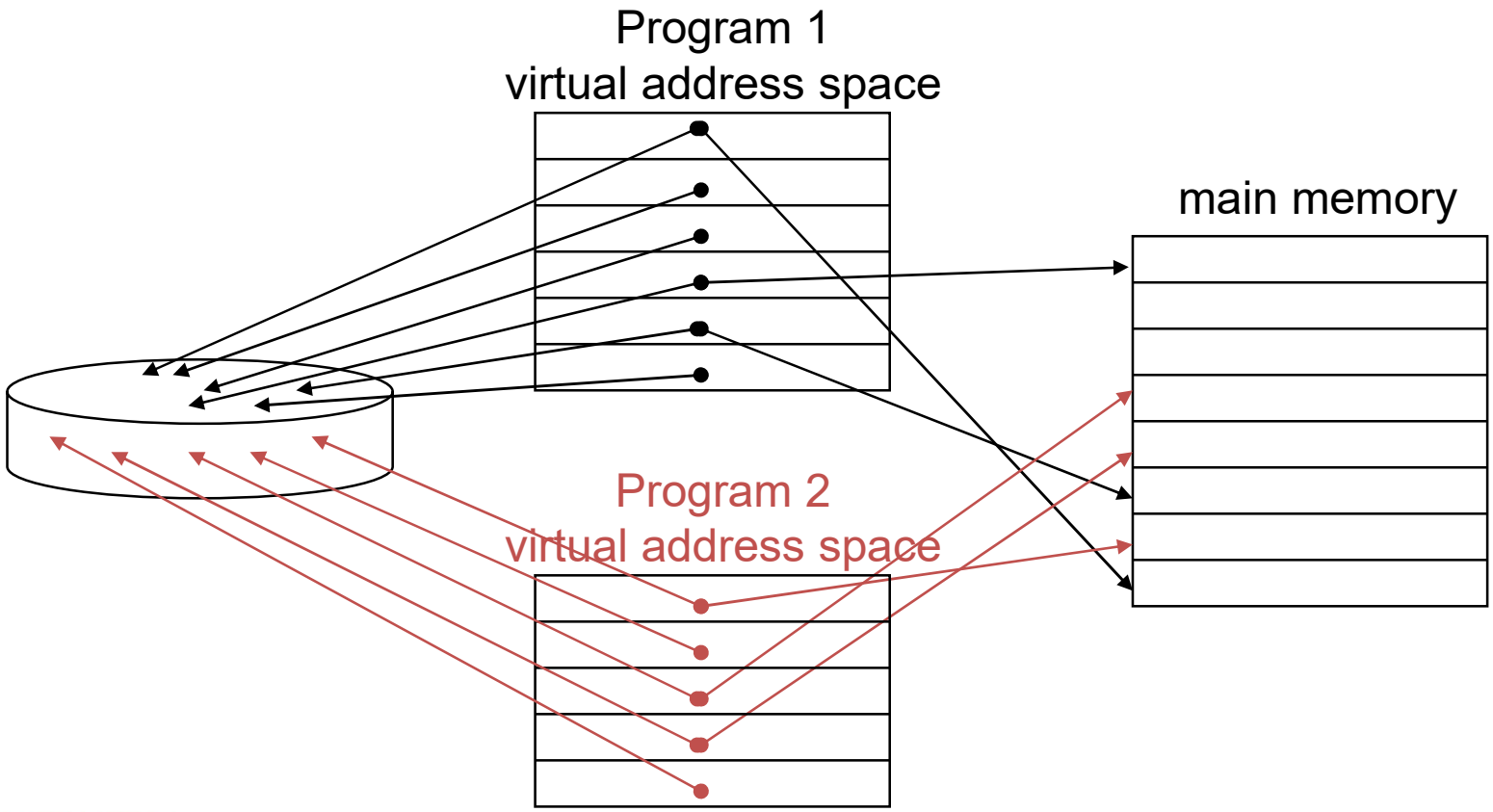
TÉCNICO LISBOA

Virtual Memory

- Use main memory as a “cache” for secondary (disk) storage
  - Managed jointly by CPU hardware and the operating system (OS)
- Each program is compiled into its own address space – a “virtual” address space
- Programs share main memory
  - Each gets a private physical address space holding its frequently used code and data
  - Protected from other programs
- CPU and OS translate virtual addresses to physical addresses
  - VM “block” is called a page
  - VM translation “miss” is called a page fault

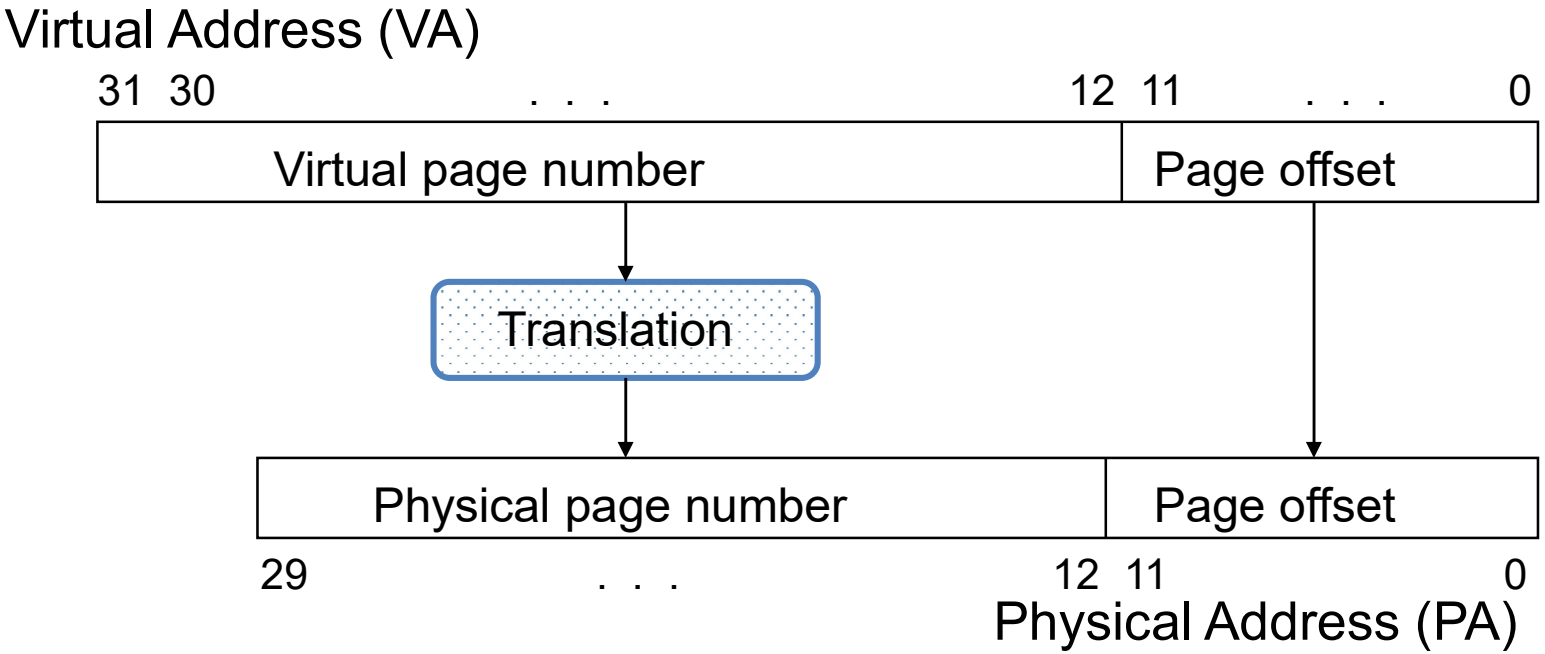
Two Programs Sharing Physical Memory

- A program's address space is divided into pages (all one fixed size) or segments (variable sizes)
  - The starting location of each page (either in main memory or in secondary memory) is contained in the program's page table



Address Translation

- A virtual address is translated to a physical address by a combination of hardware and software



- So each memory request *first* requires an address translation from the virtual space to the physical space
  - A virtual memory miss (i.e., when the page is not in physical memory) is called a page fault

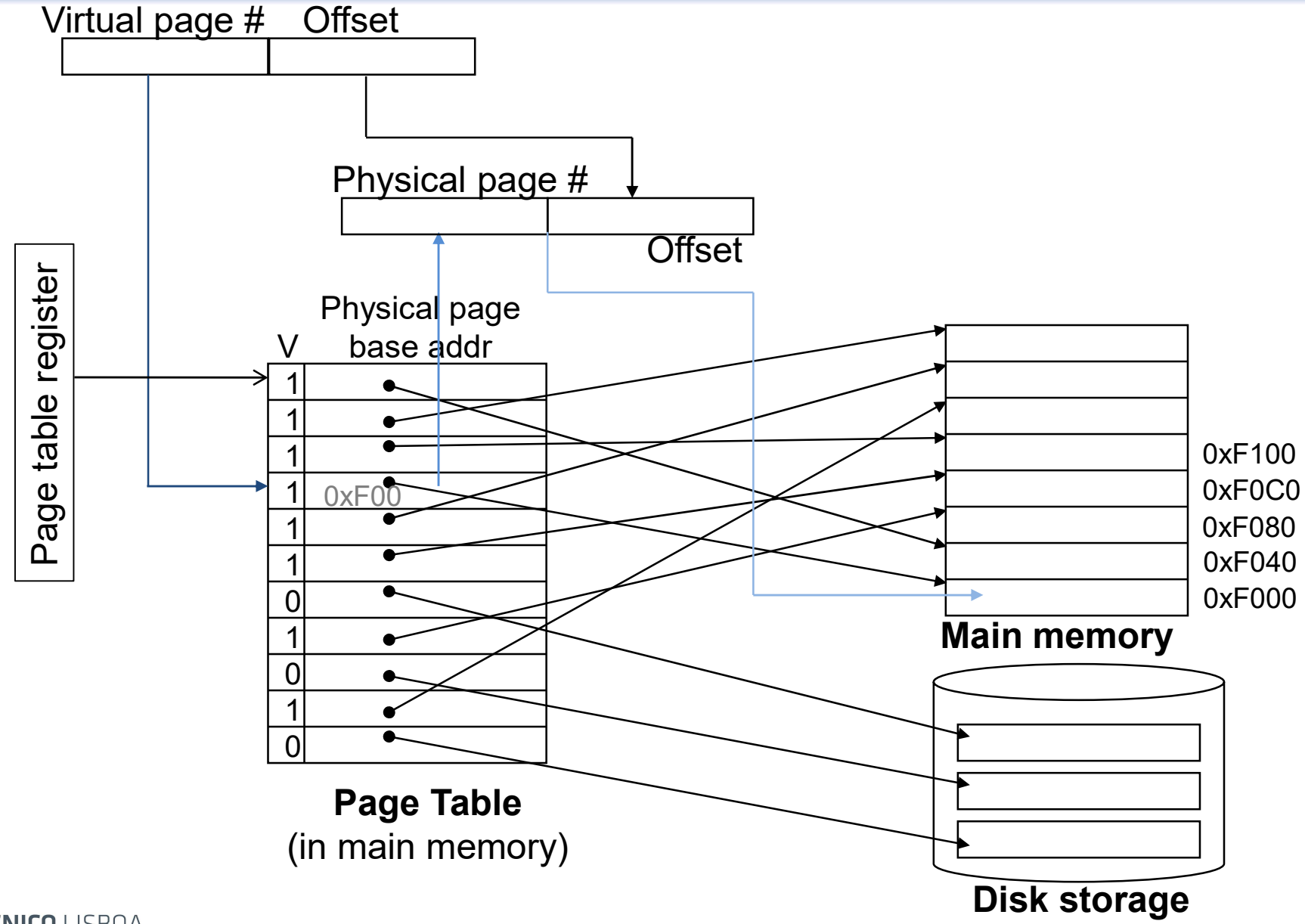
Address Translation

- Just as cache accommodates a subset of primary memory positions, the primary memory accommodates a subset of virtual memory positions.
  - Block = page
  - Dimension:

    Usually large, to increase the efficiency when acceding the disk (also reduces the dimension of translation tables). However, the bigger it is, the greatest is the potential memory waste (on average, 50% of the page dimension). Typical values are 4k and 8k bytes.
  - Where can a given block be placed in memory (i.e., set-associativity):

    In any place of the memory (full associativity).

Translation Using a Page Table



Page Tables

- Stores placement information
  - Array of page table entries, indexed by virtual page number
  - Page table register in CPU points to page table in physical memory
- If page is present in memory
  - PTE (Page Table Entry) stores the physical page number
  - Plus other status bits (referenced, dirty, …)
- If page is not present
  - PTE can refer to location in swap space on disk

Replacement and Writes

- To reduce page fault rate, prefer least-recently used (LRU) replacement
  - Reference bit (aka use bit) in PTE
    - Set to 1 on access to page
    - Periodically cleared to 0 by OS
  - A page with reference bit = 0 has not been used recently
- Disk writes take millions of cycles
  - Block at once, not individual locations
  - Write through is impractical
  - Use write-back
  - Dirty bit in PTE set when page is written

Pages Table Dimension

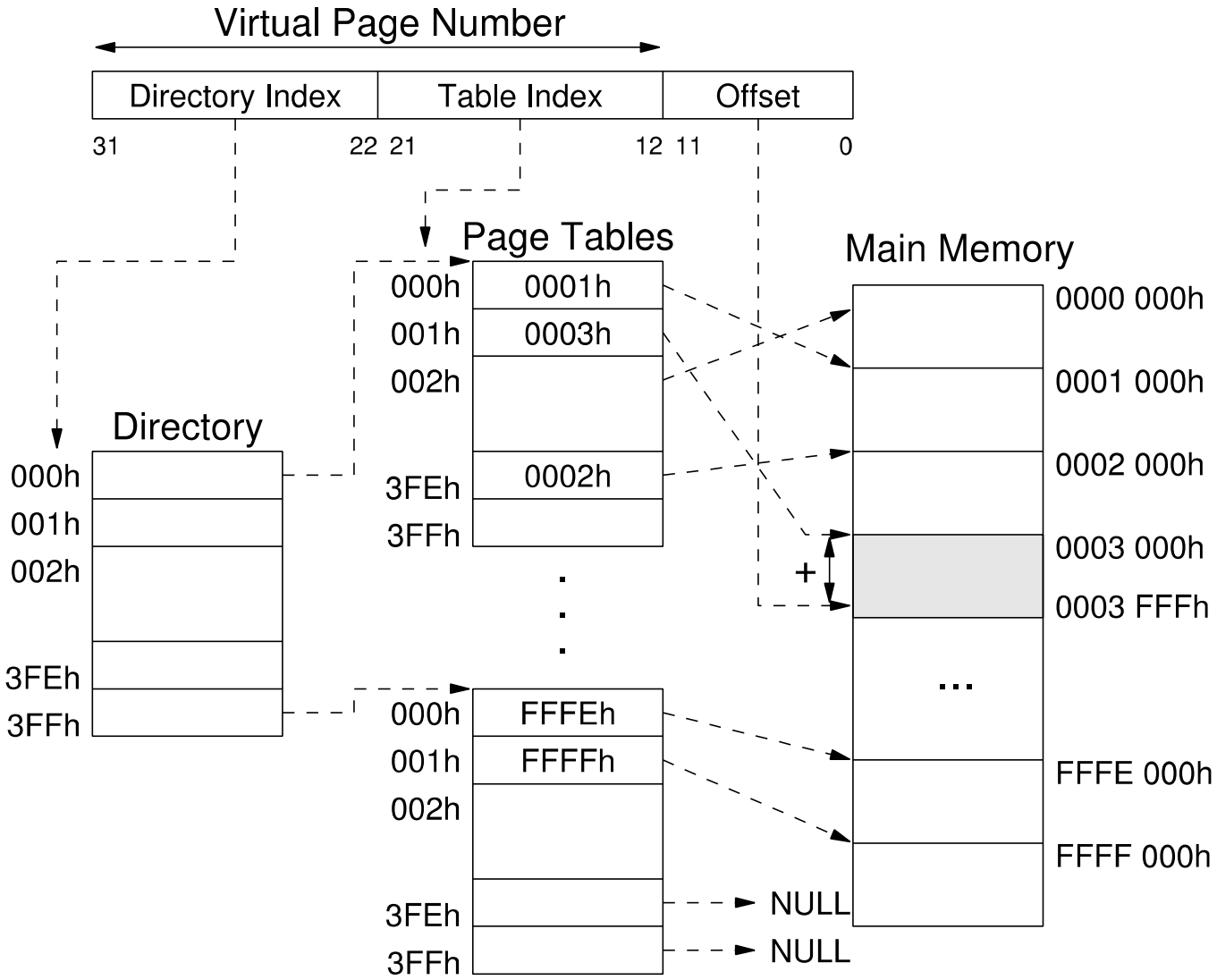
- One common problem in paging systems is the dimension of the page table that is required to translate the addresses.
  - Must be allocated in a contiguous region at physical memory.
- Example:
  - Virtual space with 2^{32} bytes,
  - Pages with 4k(= 2^{12}) bytes,
  - Descriptor with D = 4 bytes:
    - Table with 2^{20} entries (4M bytes).

Hierarchical Page Tables

- One common solution to this problem is to implement the translation with a hierarchy of translation tables.

- Virtual address is seen as:

Hierarchical Page Tables



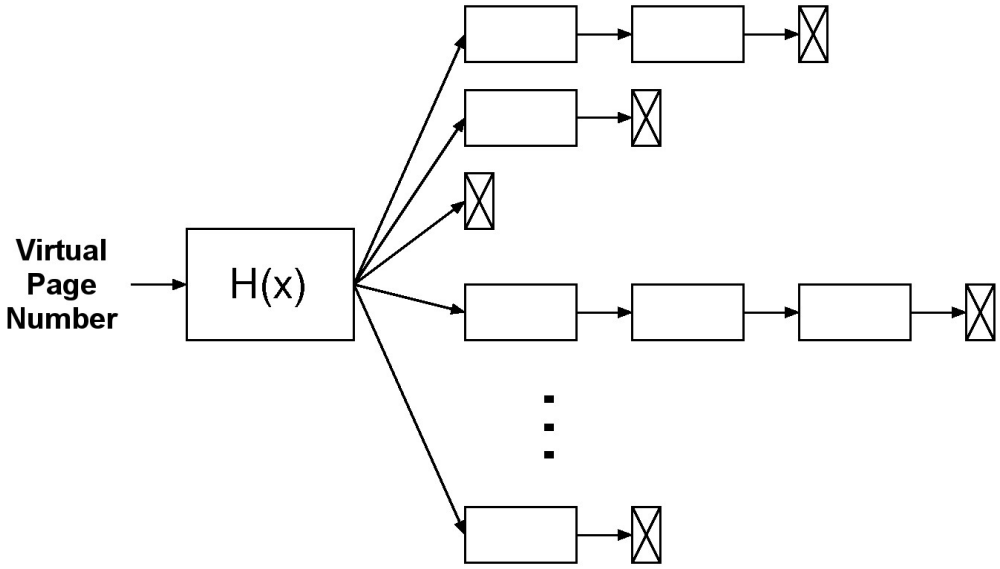
Hierarchical Page Tables

- How much space is required to store the tables?
- Example:
  - 2 levels, virtual address with 2^{32} bytes and pages with 4k(= 2^{12}) bytes

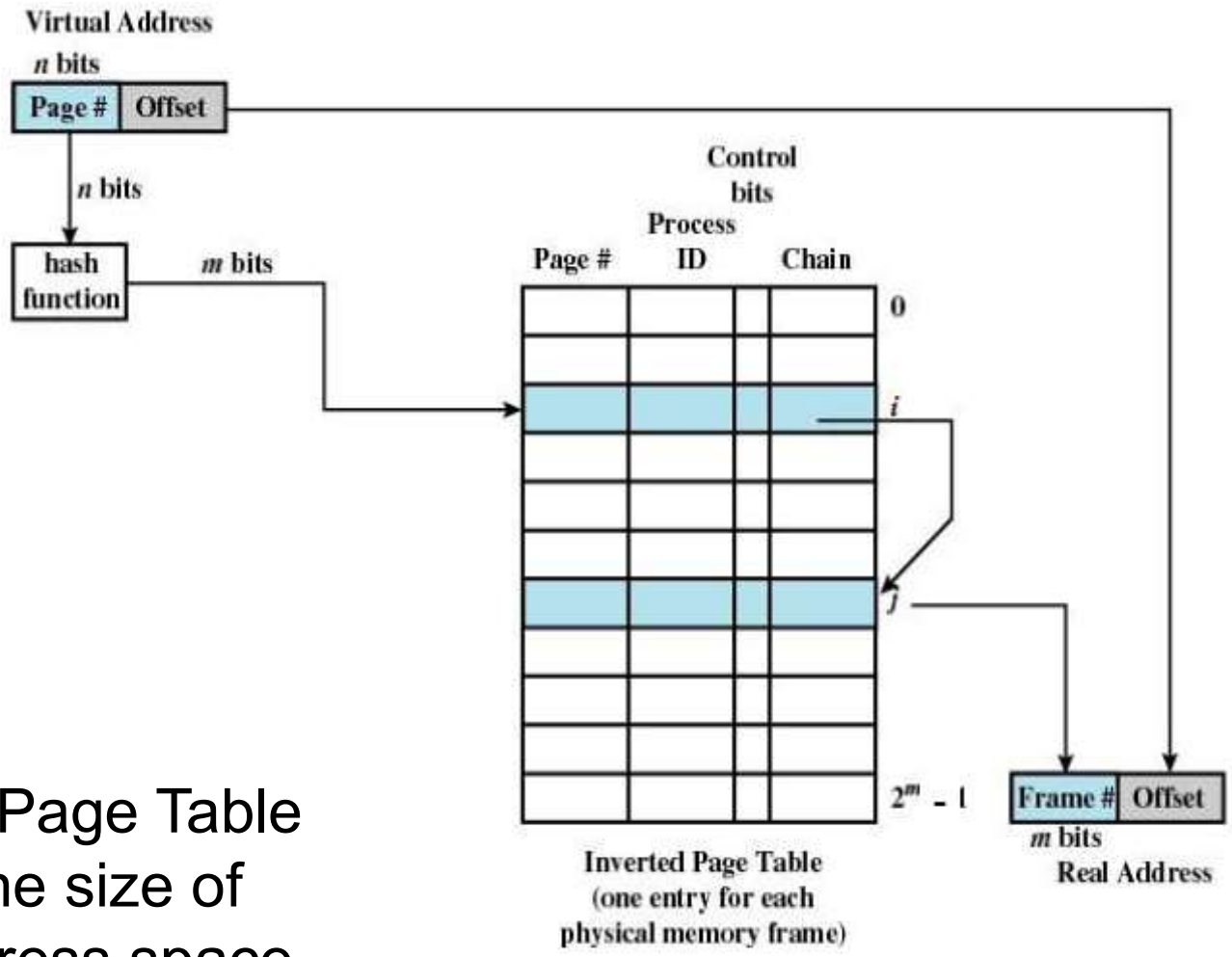
- Directory has 1024 entries. Each level 2 table has 1024 entries. The total is 1M entries, corresponding to 4M bytes, just as before!
  - Only the directory must be always in memory
    - the higher level tables may be at disk!
  - Usually, it is adopted:
    - Page tables with the same dimension as the pages;
    - Only the pages under use have to be in primary memory;
    - Only the required pages are instantiated.
- The levels can be divided into different size indexes.

Inverted Tables

- Address translation based in hash tables:
  - A given hash function H(x) is applied to the virtual address to find out a particular queue of descriptors composed by pairs “virtual page - physical page”, whose corresponding virtual addresses lead to the same value of the hash function H(x)⟶collisions.
  - The required physical address may (or may not) be present in that queue of descriptors.



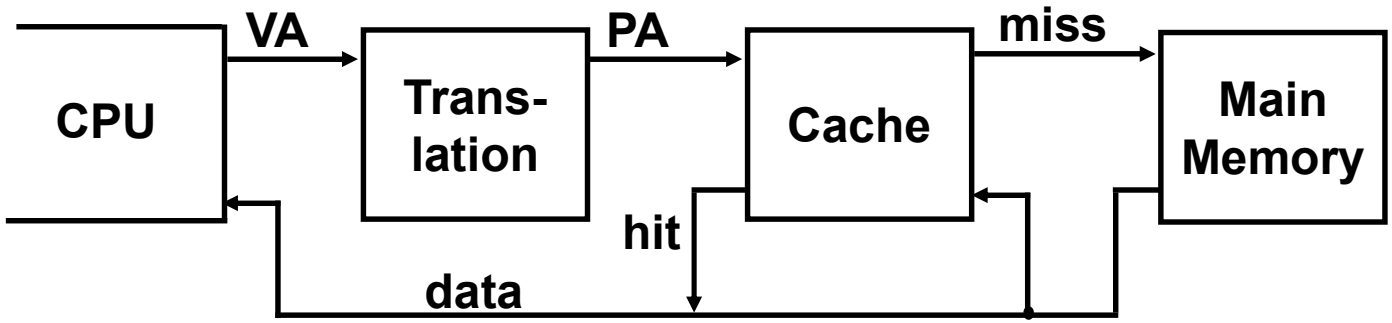
Address Translation with Inverted Tables



Size of Inverted Page Table proportional to the size of the physical address space.

Virtual Addressing with a Cache

- Thus it takes at least an *extra* memory access to translate a VA to a PA



- This makes memory accesses even more expensive
- The hardware fix is to use a Translation Lookaside Buffer (TLB)
  - a small cache that keeps track of recently used address mappings to avoid having to do a page table lookup

Next Class

- Translation Lookaside Buffer (TLB)
- Integrated Operation of the Memory System

Virtual Memory

Computer Organization

Monday, 30 September 2024

Many slides adapted from:
Computer Organization and Design,
Patterson & Hennessy
5th Edition, © 2014, MK
and from Prof. Mary Jane Irwin, PSU

TÉCNICO LISBOA