Embedded Systems Design
Lecture 9

Yongsoo Joo

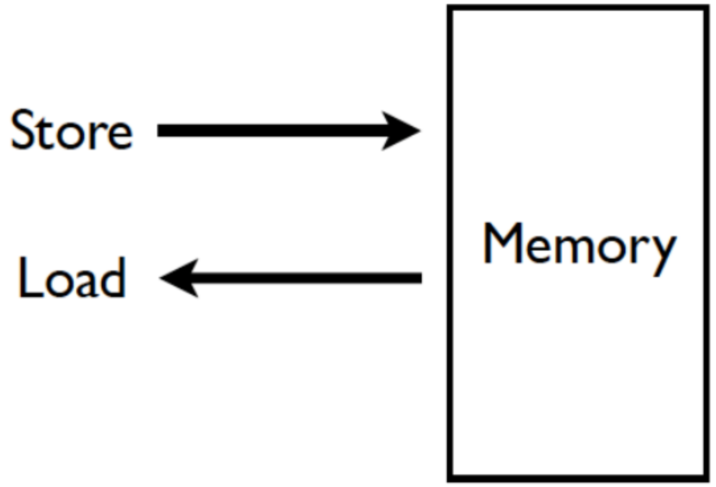
KESL
Kookmin Univ. Embedded System Lab

VM for Embedded Systems

- Virtual memory
  - Not an issue for embedded systems
    - A single dedicated application without OS support
  - Things changed
    - Many of modern embedded systems rely on OS

Memory Management

- Goals
  - Convenient abstraction for programming
  - Provide isolation between different processes
  - Allocate scarce physical memory resources across processes
    - Especially important when memory is heavily contended for
- Mechanisms
  - Virtual address translation
  - Paging and TLBs
  - Page table management
- Policies
  - Page replacement policies



Programmer's view of memory

Virtual Memory

- What is VM?
  - The basic abstraction provided by the OS for memory management
  - Enables programs to be executed without requiring their entire address space to be resident in physical memory
    - Called "Demand paging"
- Observation
  - Many programs don't use all of their code or data
    - Ex) Branches never taken, variables never accessed, etc.
  - No need to allocate memory for it until it's used

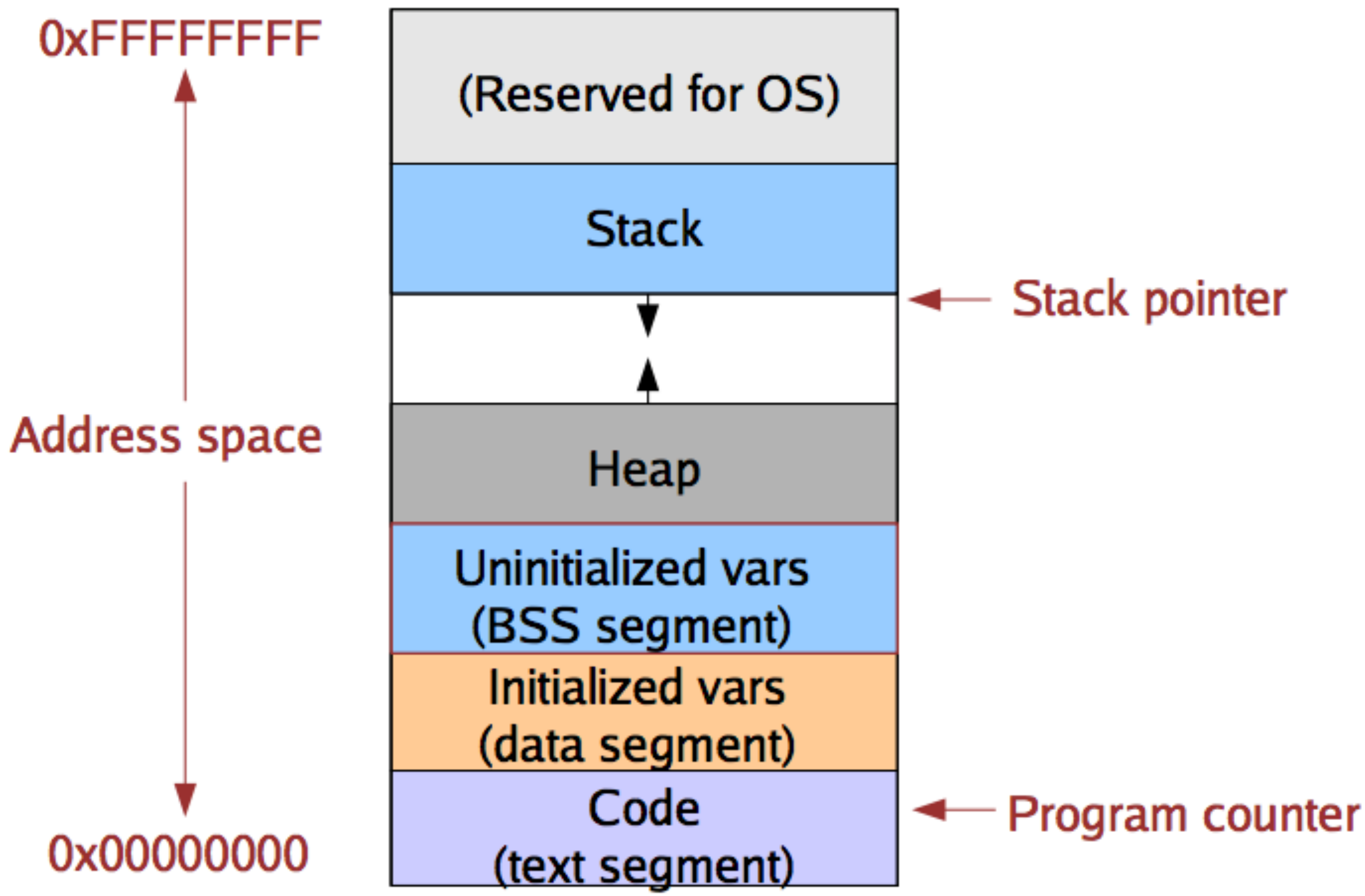
Virtual Memory

- Functions
  - OS uses VM to adjust amount **of physical memory** allocated **to each process** based on its run-time behavior
  - VM also provides isolation among processes
  - Each process has its own isolated address space
  - One process cannot access memory addresses in others
- Implementation
  - HW support: address translation
    - Memory management unit (MMU), translation lookaside buffer (TLB)
  - OS support: mapping between VA-to-PA
    - Page fault handler, page table

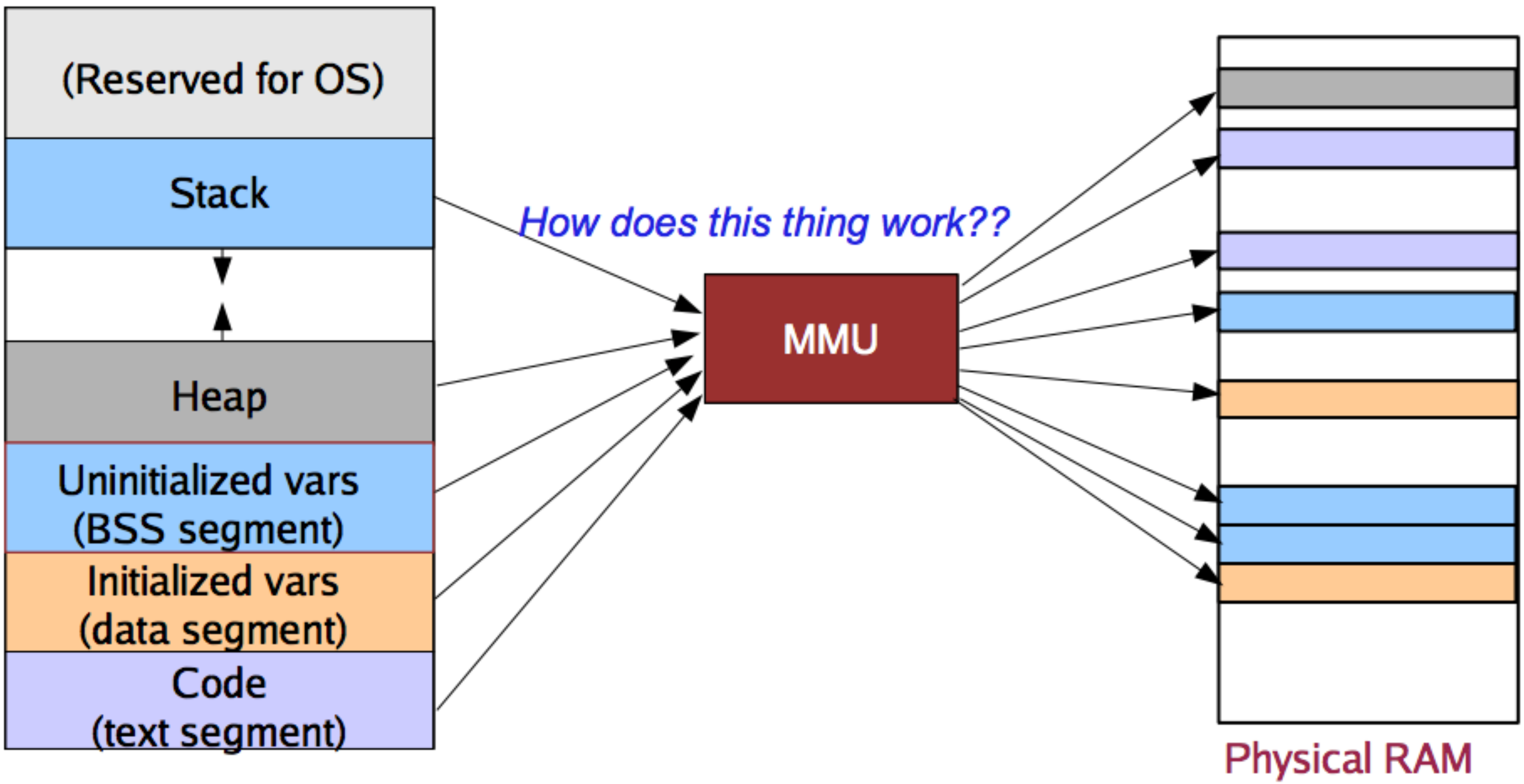
Virtual Address (VA)

- VA: a memory address that a process uses to access its own memory
  - VA != PA (physical address)
  - PA = MMU(VA)
  - VA-to-PA mapping
    - Determined by OS

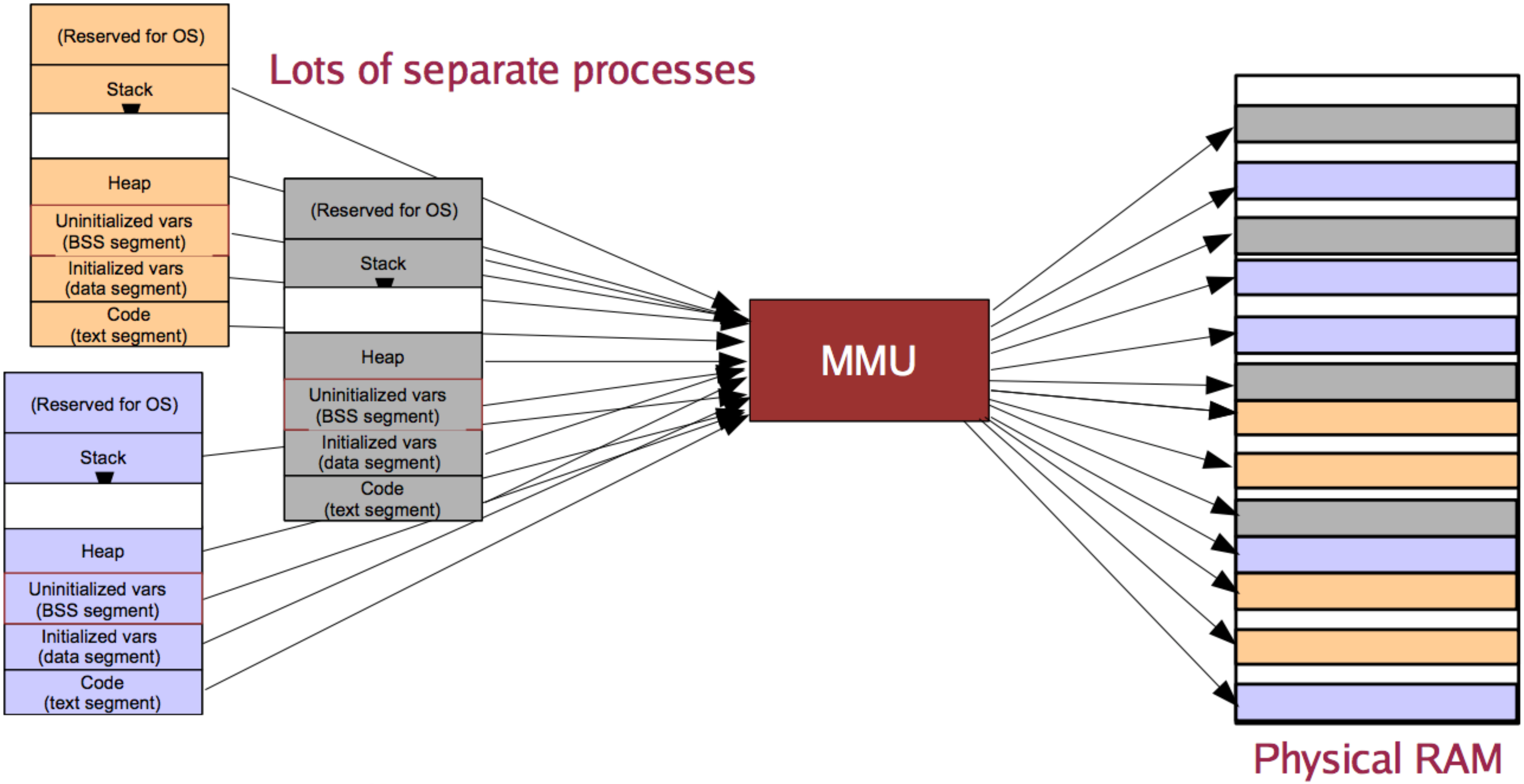
Virtual Address (VA)



Virtual Address (VA)



Application Perspective

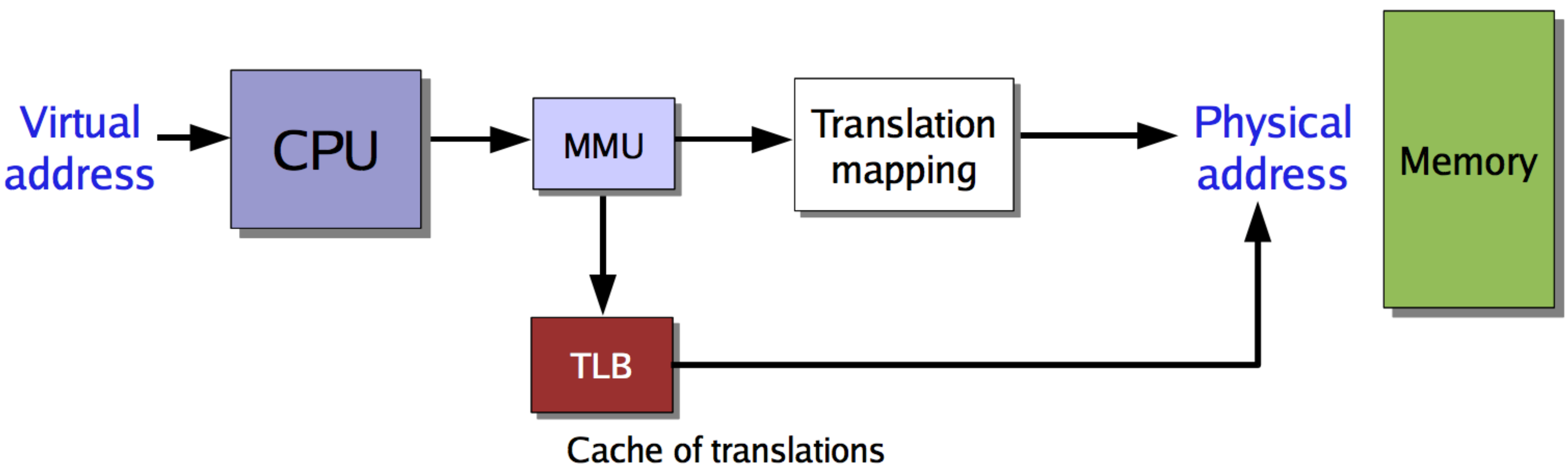


Virtual Address

- Isolation
  - VA in one process refer to different physical memory than virtual addresses in another
    - Exception: shared memory regions btw. processes
- Relocation
  - A program does not need to know which PA it will use when it's run
  - Compilers generate relocatable code

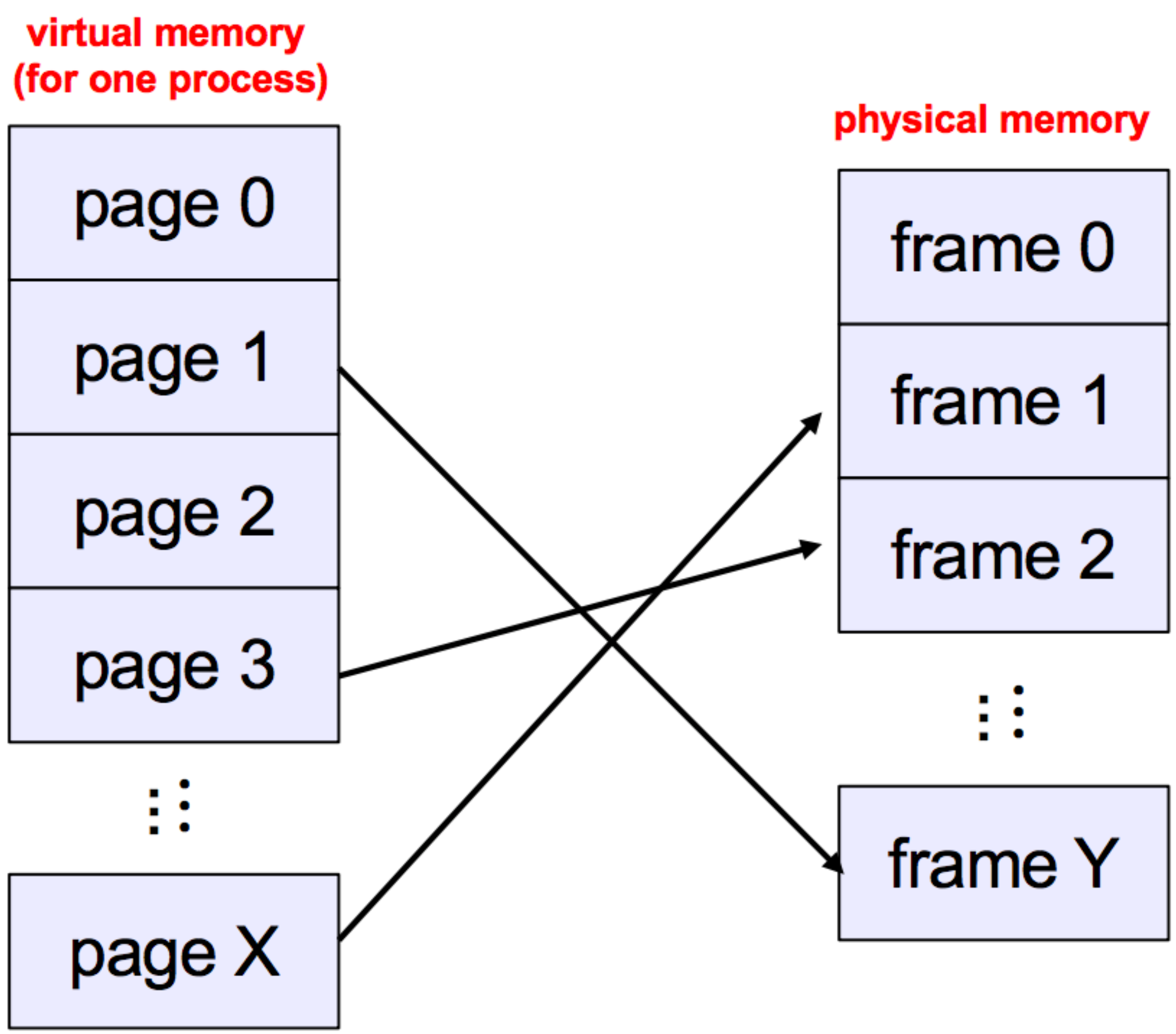
MMU and TLB

- MMU (memory management unit)
  - HW that translates a VA to a PA
  - PA=MMU(VA)
- TLB (translation lookaside buffer)
  - Cache for MMU V-to-P address translations



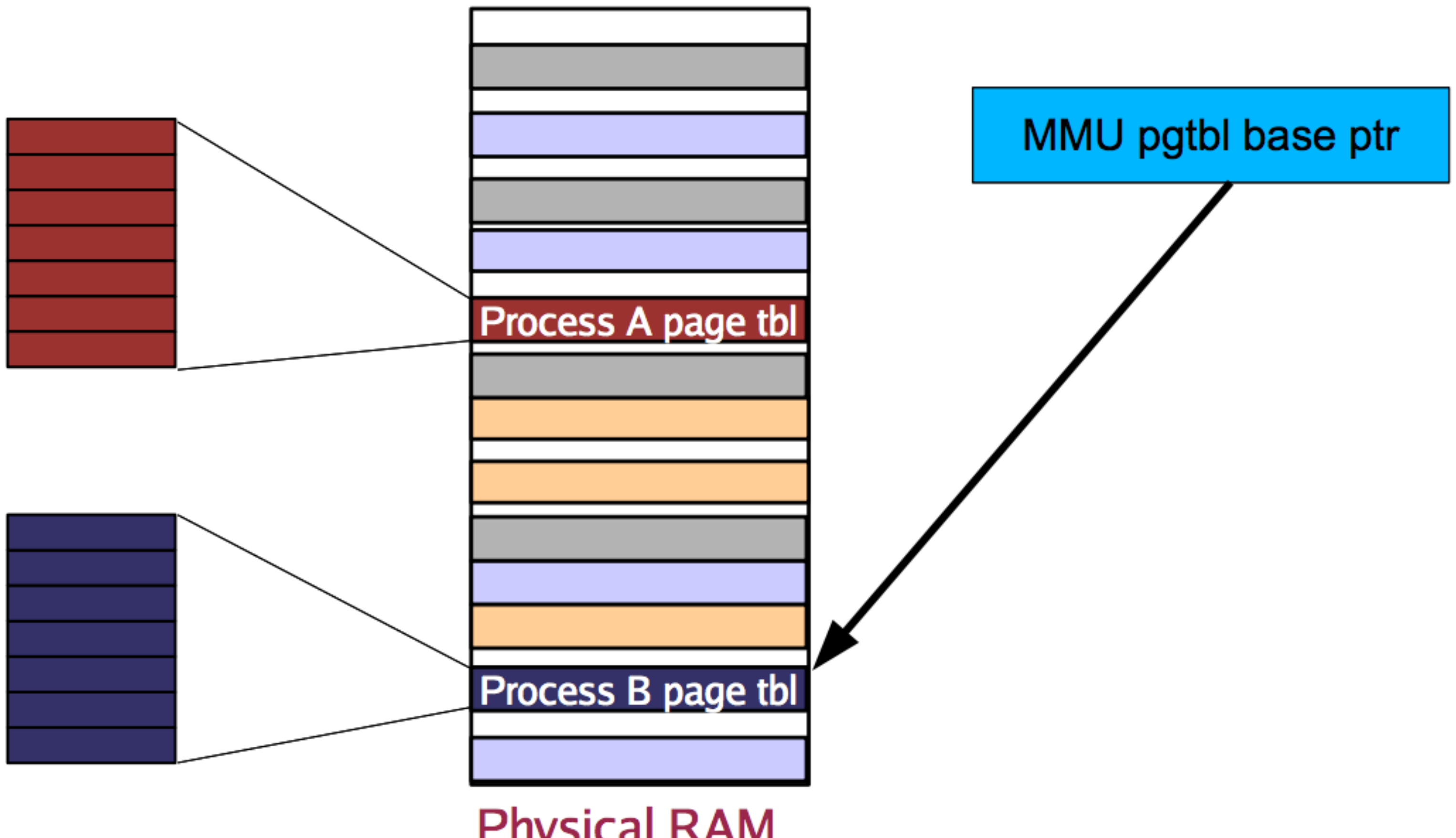
Paging

- Virtual memory is divided into fixed-size chunks called pages
- Physical memory is divided into page frames



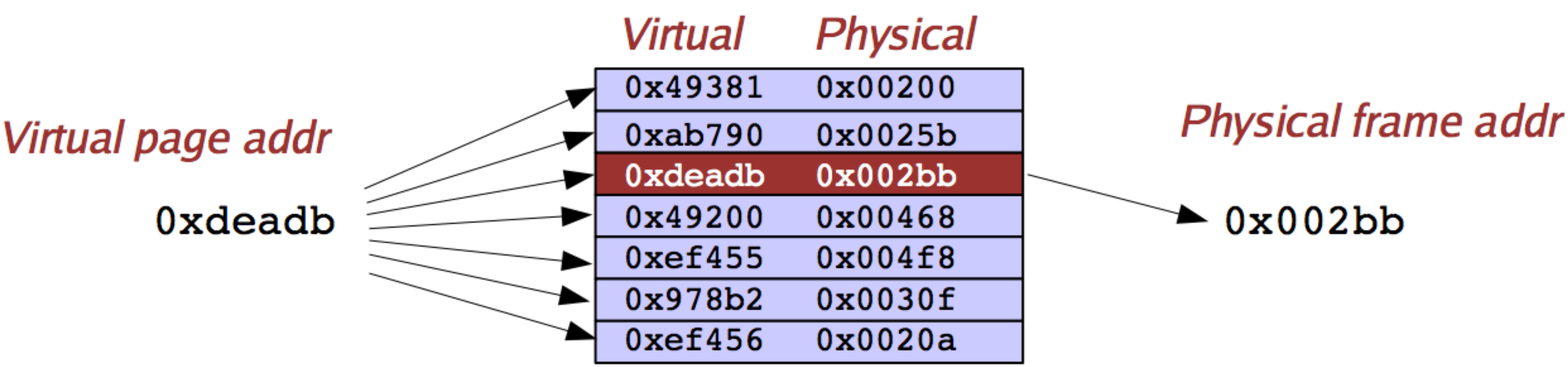
Page Tables

- Store the virtual-to-physical address mappings
- Location: in memory
- Way to access
  - MMU has a special register called the page table base register
  - It points to the physical memory address of the top of the page table for the currently-running process



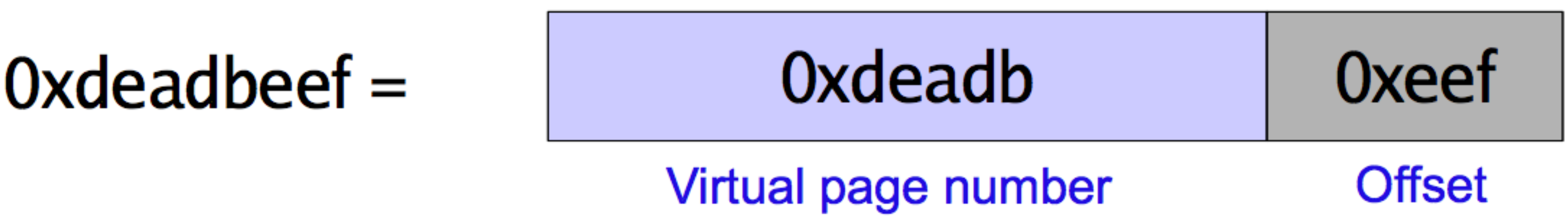
The TLB

- MMU overhead
  - Each memory access requires an additional memory access to the page table (100% overhead)
- Solution: Translation Lookaside buffer (TLB)
  - Very fast (but small) cache directly on the CPU
  - Caches most recent v-to-p address translations
  - A TLB miss requires access to the page table on the main memory

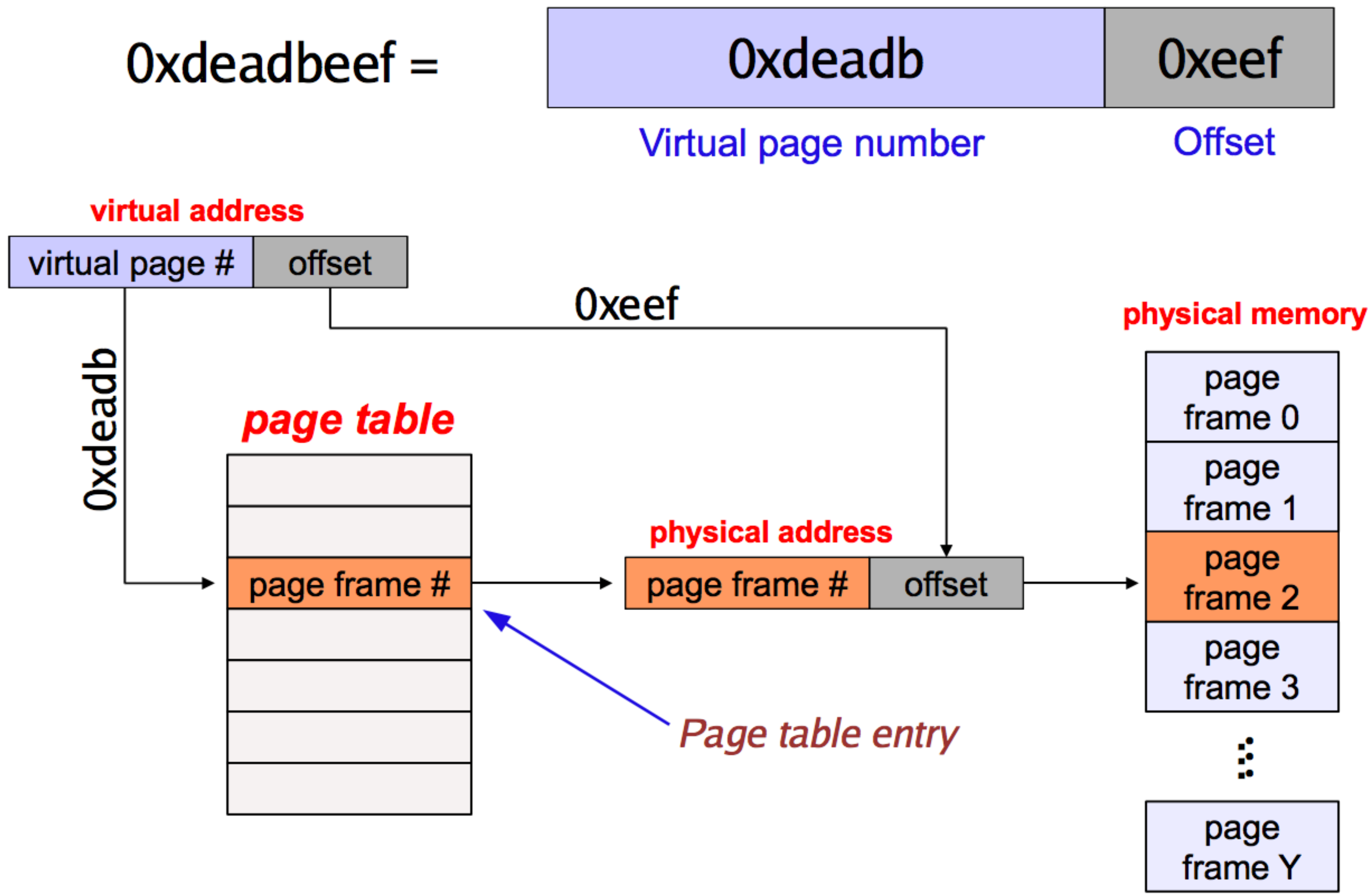


Virtual Address Translation

- Performed by MMU
  - VA is broken into Virtual page number and an offset
  - Mapping from a virtual page to a physical frame provided by a page table



Virtual Address Translation



KESL
Kookmin Univ. Embedded Systems Lab

Page Table Entries (PTEs)

1	1	1	2	20
V	M	R	prot	page frame number

- Valid bit (V): whether the corresponding page is in memory
- Modify bit (M): indicates whether a page is "dirty"
- Reference bit (R): whether a page has been accessed
- Protection bits: specify if the page is readable, writable, or executable
- Page frame number: physical location of page in main memory
  - PFN+OFFSET=physical memory address

Reference

http://www.eecs.harvard.edu/~mdw/course/cs161/notes/vm.pdf

https://cs.umd.edu/class/spring2015/cmsc411-0201/lectures/lecture14_virtual_memory_2.pdf

http://codex.cs.yale.edu/avi/os-book/OS9/slide-dir/PPT-dir/ch9.ppt