

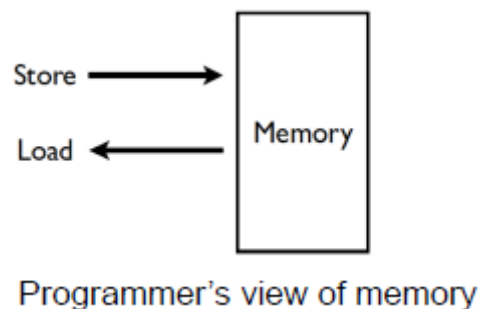
Virtual Memory

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
- 자원(memory)을 요청하는 client가 많을 수 있다
- Virtual Memory로 인해 **실시간성**에 심각한 악영향을 끼친다

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



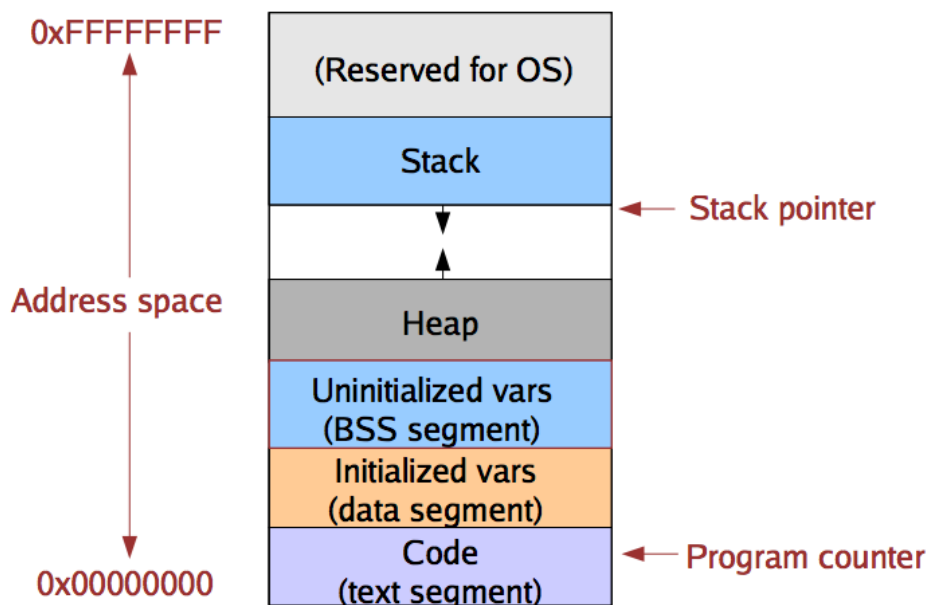
Virtual Memory

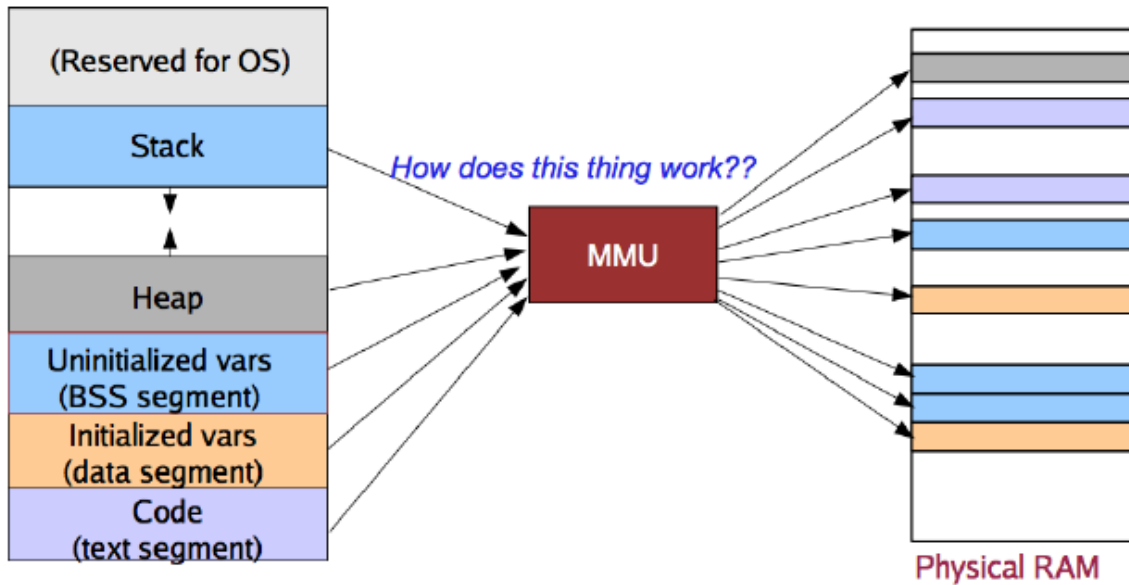
- VM?
 - The basic abstraction provided by the OS for memory management
 - Enables programs to be executed without requiring their entire address space to be resident in physical memory
 - called "Demand Paging"
- Observation
 - Many programs don't use all of their code or data(효율적이지 못하다)
 - ex) branches never taken, variables never accessed

- No need to allocate memory for it until it's used
- 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
 - 하드웨어의 도움을 많이 받음
 - Each process has its own isolated address space
 - One process cannot access memory addresses in others
- Implementation
 - HW support : address translation
 - MMU(Memory Management Unit), TLB(Translation Lookaside Buffer)
 - MMU가 없으면 virtual memory 구축안됨
 - 간단하고 빈도가 잦은 작업
 - OS support : mapping between **VA-to-PA**
 - Page fault handler, page table
 - handling하는 function 제공(CPU가 아닌 Software 영역)
 - 복잡하고 빈도가 낮은 작업

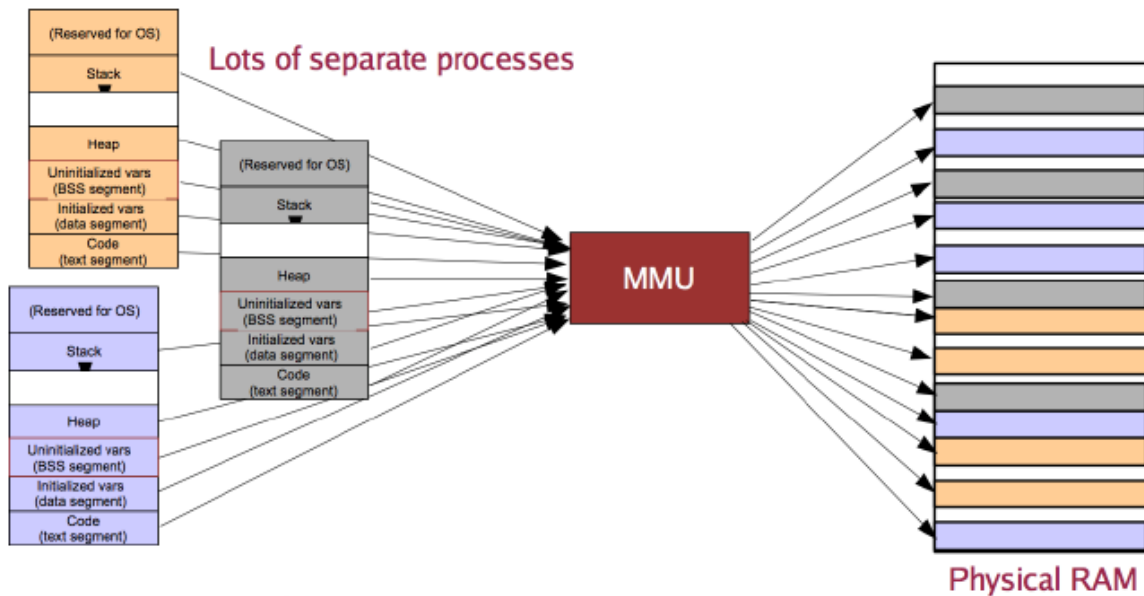
Virtual Address(VA)

- VA : a memory address that a process uses to access its own memory
 - $VA \neq PA$ (Physical Address)
 - $PA = MMU(VA)$
 - MMU가 VA를 받으면 PA로 바꿔준다
 - VA-to-PA mapping
 - Determined by OS





- 무조건 MMU를 통과해야함 : isolation이 가능한 이유

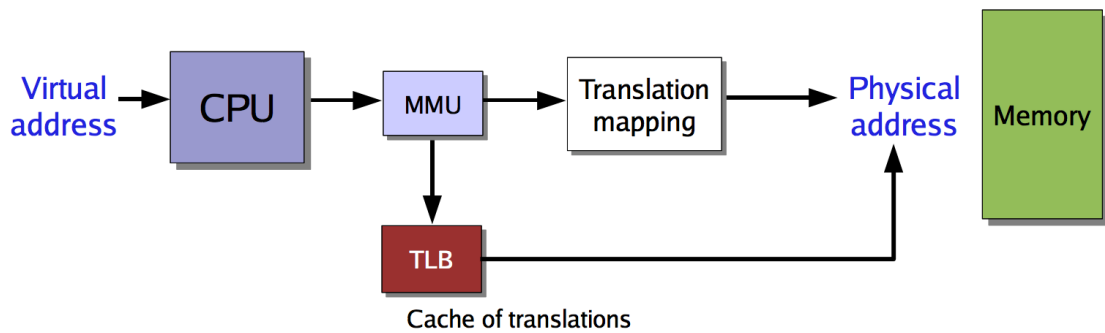


- Application Perspective
- Isolation
 - VA in one process refer to different physical memory than virtual addresses in another
 - Exception : **shared memory regions** between processes
- Relocation
 - A program does **not need to know which PA** it will use when it's run
 - Compilers generate relocatable code
 - 최종적으로 OS에서 프로세스를 실행시킬 때 주소가 정해진다

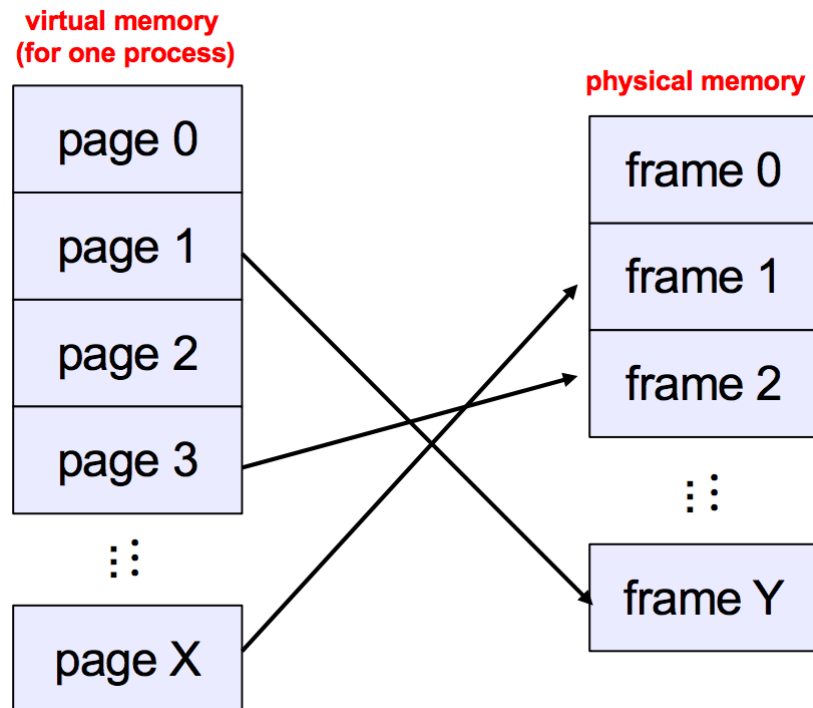
MMU and TLB

- MMU
 - HW that translates a VA to a PA
 - $PA = MMU(VA)$
- TLB
 - Cache for MMU V-to-P address translations

- locality 발생, 아주 작은 cache의 역할



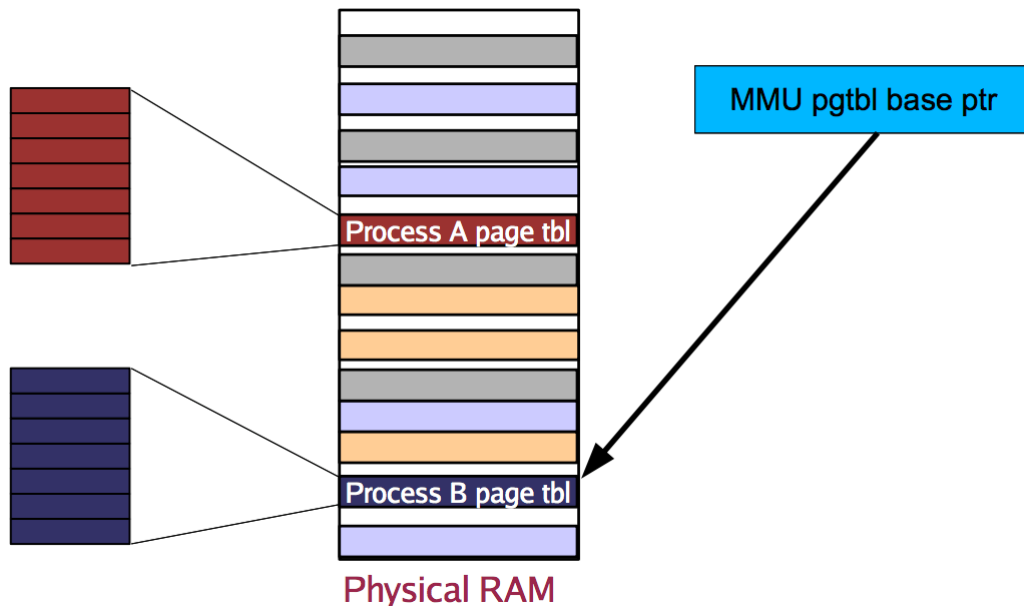
Paging



- Virtual memory is divided into fixed-size chunks called pages
- Physical memory is divided into page frames
 - Page to Frame mapping : MMU가 하는일
- 4 KB chunk

Page Tables

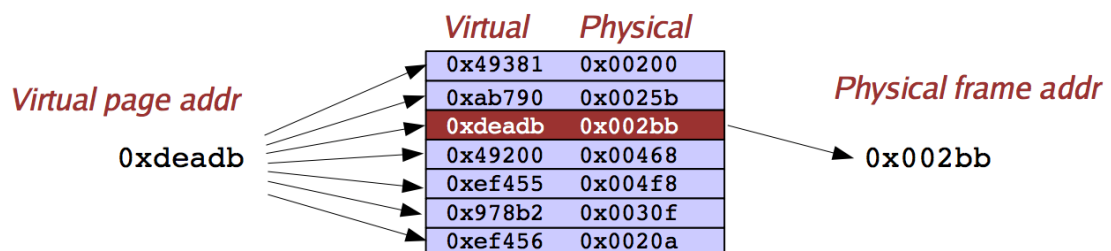
- Store the virtual-to-physical address mappings
- Location : in **main memory**
- Way to access
 - MMU has a **special register called the page table base register**(inside cpu)
 - bus의 master 역할을 할 수 있다(bus 입장에서는 똑같이 CPU가 쓰는걸로 인식)
 - It points to the physical memory address of the top of the page table for the currently-running process



- offset을 표현하기 위해 12개의 비트(4KB 페이지 기준)가 필요하다
- 32 bit 시스템에서 앞쪽의 20개 비트는 base(여기가 virtual/physical 주소 변환이 이뤄지는 곳)
 - 주소 변환 시에는 offset 정보가 필요없기 때문에 남은 12비트에 다양한 메타데이터 보관
 - 10^6 개의 Page, page 하나 별로 4 바이트 필요
 - user process 별로 virtual address 필요? 과부하 심하다 - 100개의 프로세스 = 400MB 필요

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
 - 10% 미만으로 overhead가 떨어진다
 - miss rate를 따지면 worst case 1%, best case 0.01%



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

