

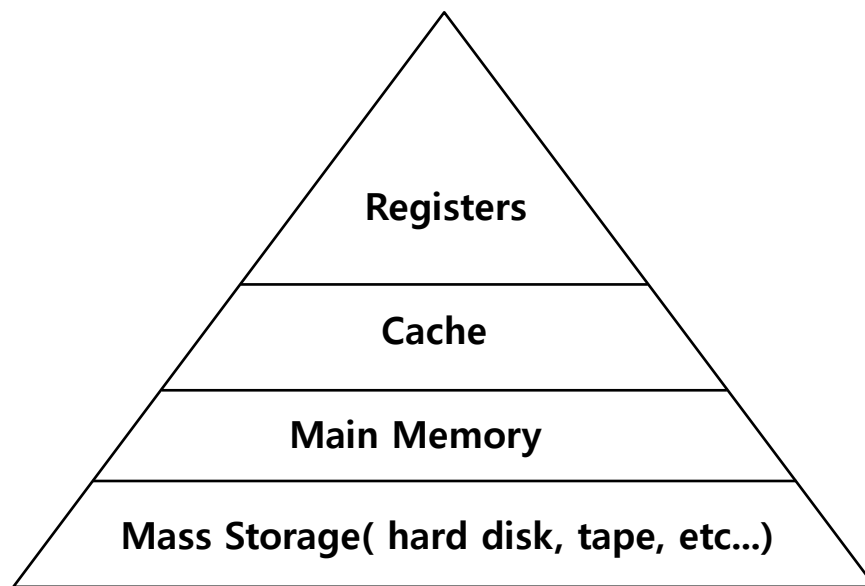
# 21. Swapping: Mechanisms

Operating System: Three Easy Pieces

---

# Beyond Physical Memory: Mechanisms

- ▣ Require an additional level in the **memory hierarchy**.
  - ◆ OS need a place to stash away portions of address space that currently aren't in great demand.
  - ◆ In modern systems, this role is usually served by a **hard disk drive**



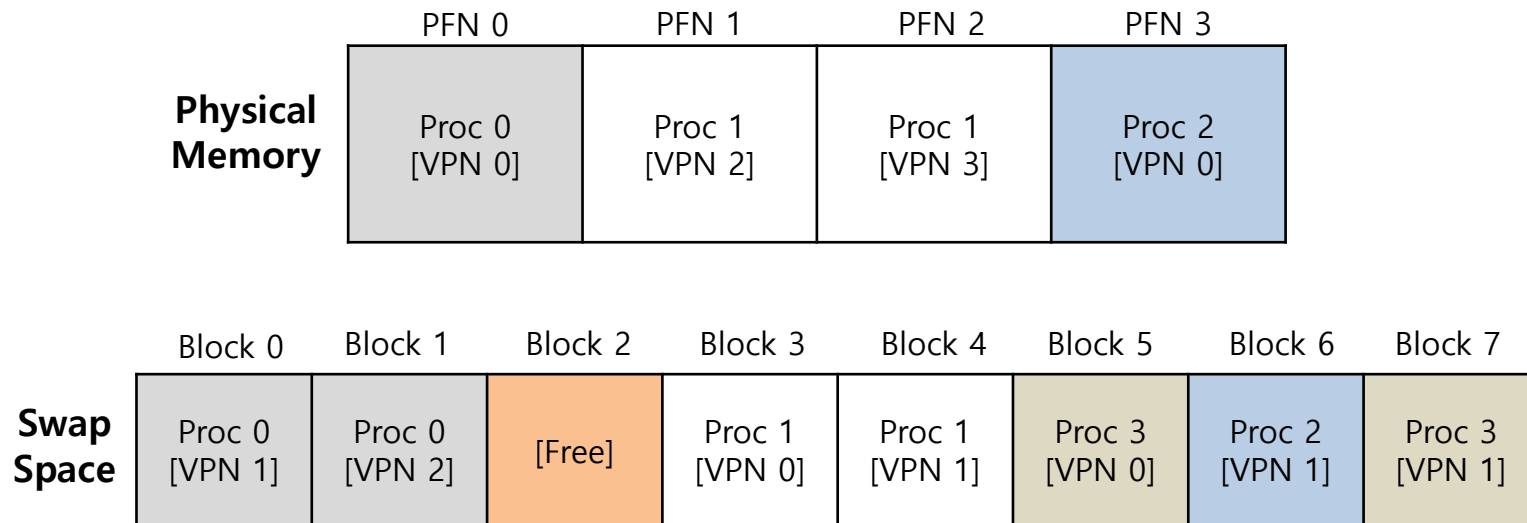
**Memory Hierarchy in modern system**

# Single large address for a process

- ▣ Always need to first arrange for the code or data to be in memory when before calling a function or accessing data.
- ▣ To Beyond just a **single process**.
  - ◆ The addition of **swap space** allows the OS to support the illusion of a large virtual memory for multiple concurrently-running process

# Swap Space

- ❑ Reserve some space on the disk for moving pages back and forth.
- ❑ OS need to remember to the swap space, in **page-sized unit**



Physical Memory and Swap Space

# Present Bit

- ▣ Add some machinery higher up in the system in order to support swapping pages to and from the disk.
  - ◆ When the hardware looks in the PTE, it may find that the page is not present in physical memory.

Value	Meaning
1	page is present in physical memory
0	The page is not in memory but rather on disk.

# What If Memory Is Full ?

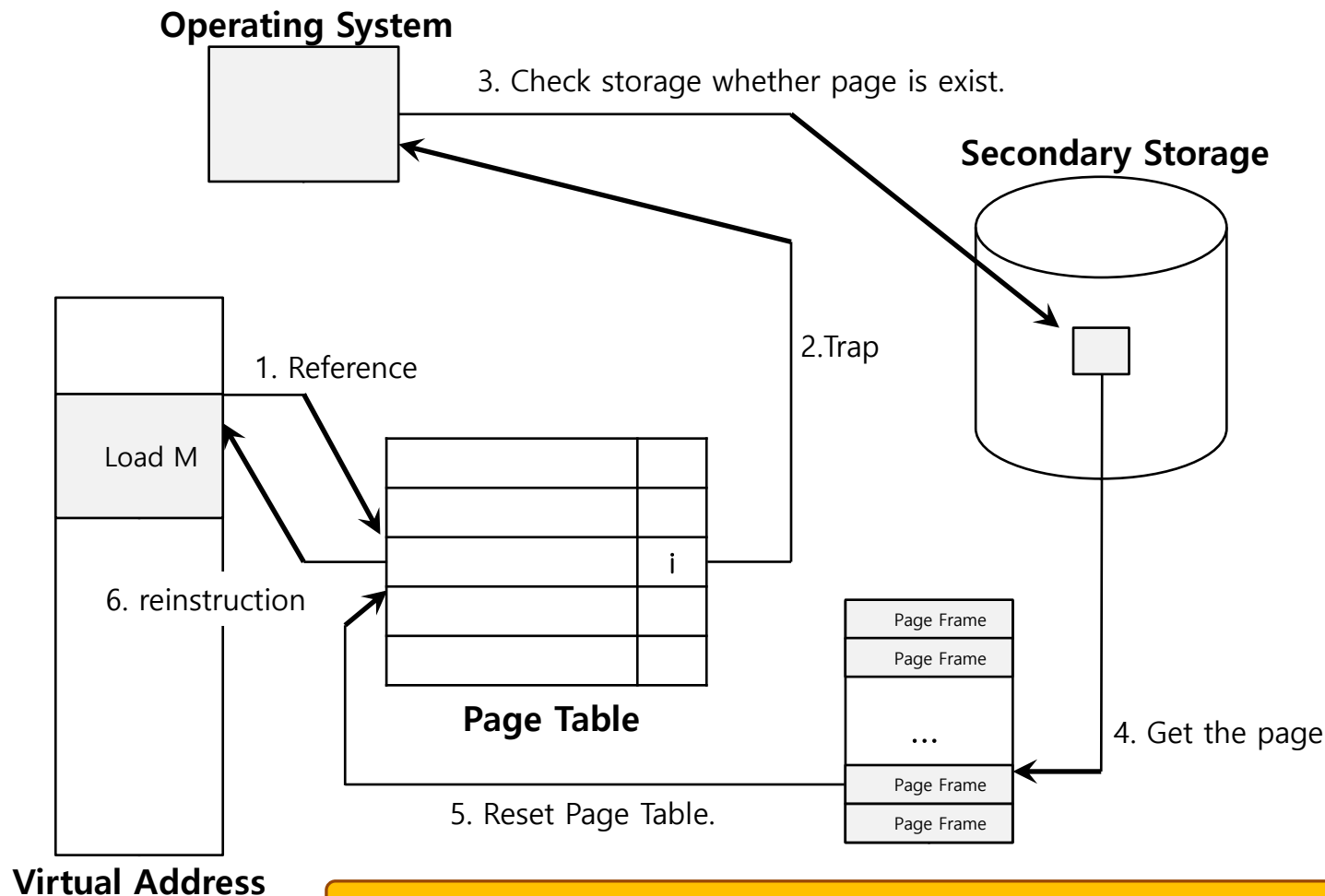
- ▣ The OS like to page out pages to make room for the new pages the OS is about to bring in.
  - ◆ The process of picking a page to kick out, or replace is known as **page-replacement** policy

# The Page Fault

- ▣ Accessing page that is **not in physical memory**.
  - ◆ If a page is not present and has been swapped disk, the OS need to swap the page into memory in order to service the page fault.

# Page Fault Control Flow

- PTE used for data such as the PFN of the page for a disk address.



When the OS receives a page fault, it looks in the PTE and issues the request to disk.



# Page Fault Control Flow – Hardware

```
1:      VPN = (VirtualAddress & VPN_MASK) >> SHIFT
2:      (Success, TlbEntry) = TLB_Lookup(VPN)
3:      if (Success == True) // TLB Hit
4:      if (CanAccess(TlbEntry.ProtectBits) == True)
5:          Offset = VirtualAddress & OFFSET_MASK
6:          PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
7:          Register = AccessMemory(PhysAddr)
8:      else RaiseException(PROTECTION_FAULT)
```

# Page Fault Control Flow – Hardware

```
9:         else // TLB Miss
10:         PTEAddr = PTBR + (VPN * sizeof(PTE))
11:         PTE = AccessMemory(PTEAddr)
12:         if (PTE.Valid == False)
13:             RaiseException(SEGMENTATION_FAULT)
14:         else
15:             if (CanAccess(PTE.ProtectBits) == False)
16:                 RaiseException(PROTECTION_FAULT)
17:             else if (PTE.Present == True)
18:                 // assuming hardware-managed TLB
19:                 TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
20:                 RetryInstruction()
21:             else if (PTE.Present == False)
22:                 RaiseException(PAGE_FAULT)
```

# Page Fault Control Flow – Software

```
1:      PFN = FindFreePhysicalPage()
2:      if (PFN == -1) // no free page found
3:          PFN = EvictPage() // run replacement algorithm
4:          DiskRead(PTE.DiskAddr, pfn) // sleep (waiting for I/O)
5:          PTE.present = True // update page table with present
6:          PTE.PFN = PFN // bit and translation (PFN)
7:          RetryInstruction() // retry instruction
```

- ◆ The OS must find a physical frame for the **soon-be-faulted-in page** to reside within.
- ◆ If there is no such page, waiting for the **replacement algorithm** to run and kick some pages out of memory.

# When Replacements Really Occur

- ▣ OS waits until memory is entirely full, and only then replaces a page to make room for some other page
  - ◆ This is a little bit unrealistic, and there are many reason for the OS to keep a small portion of memory free more proactively.
- ▣ Swap Daemon, Page Daemon
  - ◆ There are fewer than **LW pages** available, a background thread that is responsible for freeing memory runs.
  - ◆ The thread evicts pages until there are **HW pages** available.

- Disclaimer: This lecture slide set was initially developed for Operating System course in Computer Science Dept. at Hanyang University. This lecture slide set is for OSTEP book written by Remzi and Andrea at University of Wisconsin.