21. Swapping: Mechanisms

Operating System: Three Easy Pieces

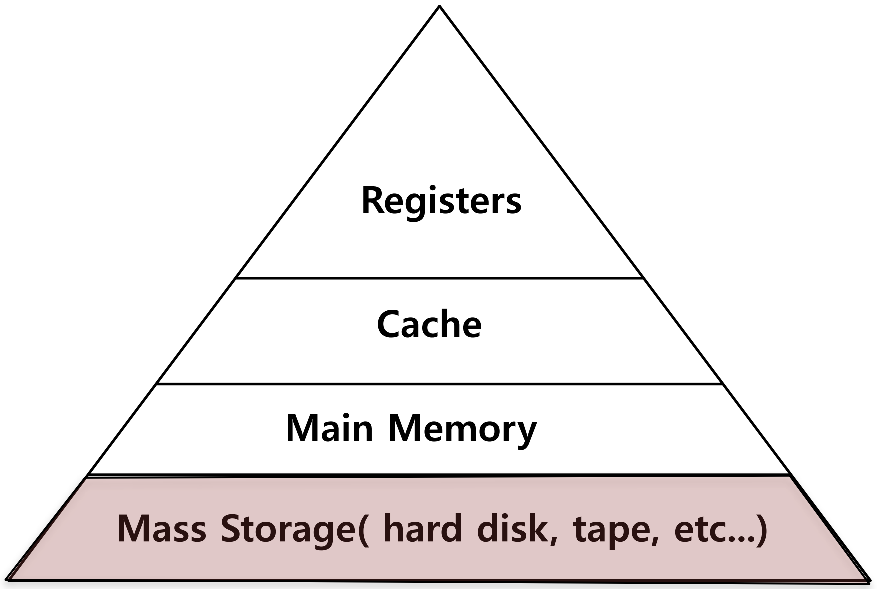
Relaxing Current Assumptions

- No more tiny address space
- No more address space “fit” into physical memory

- We need a additional level in the memory hierarchy

Beyond Physical Memory: Mechanisms

- Why 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

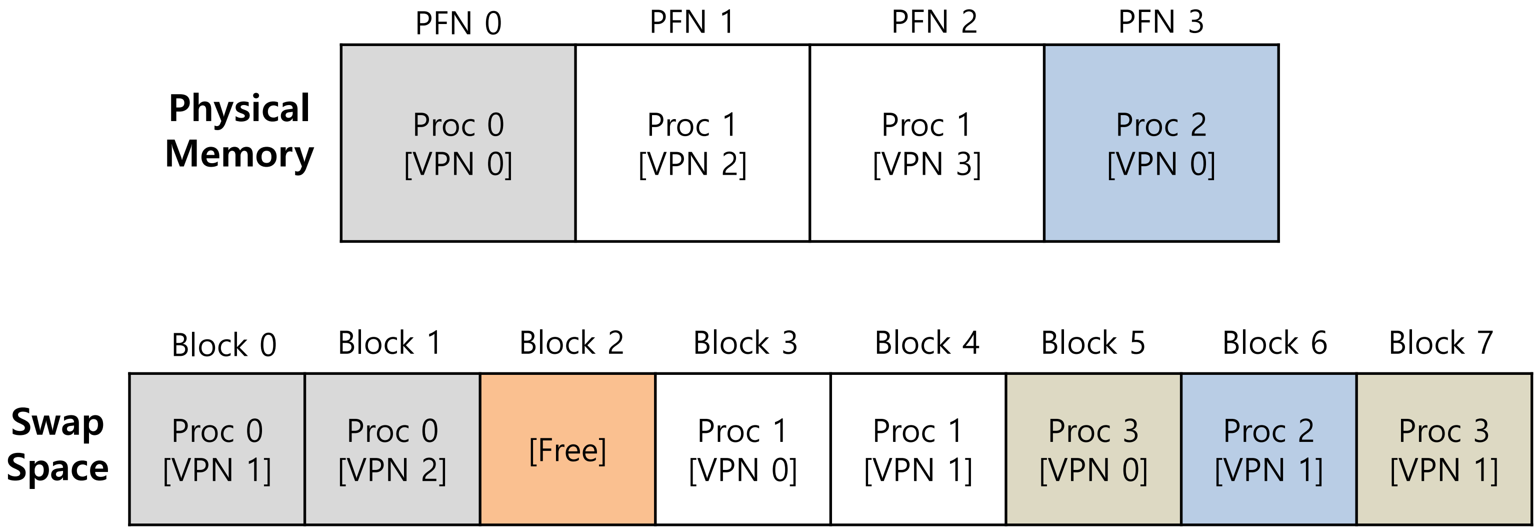
Single large address for a process

- Convenience
  - No need to first re-arrange for the code or data to be in memory when before calling a function or accessing data (because already is "there"!)
  - Otherwise (like in old days) overlays

- 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

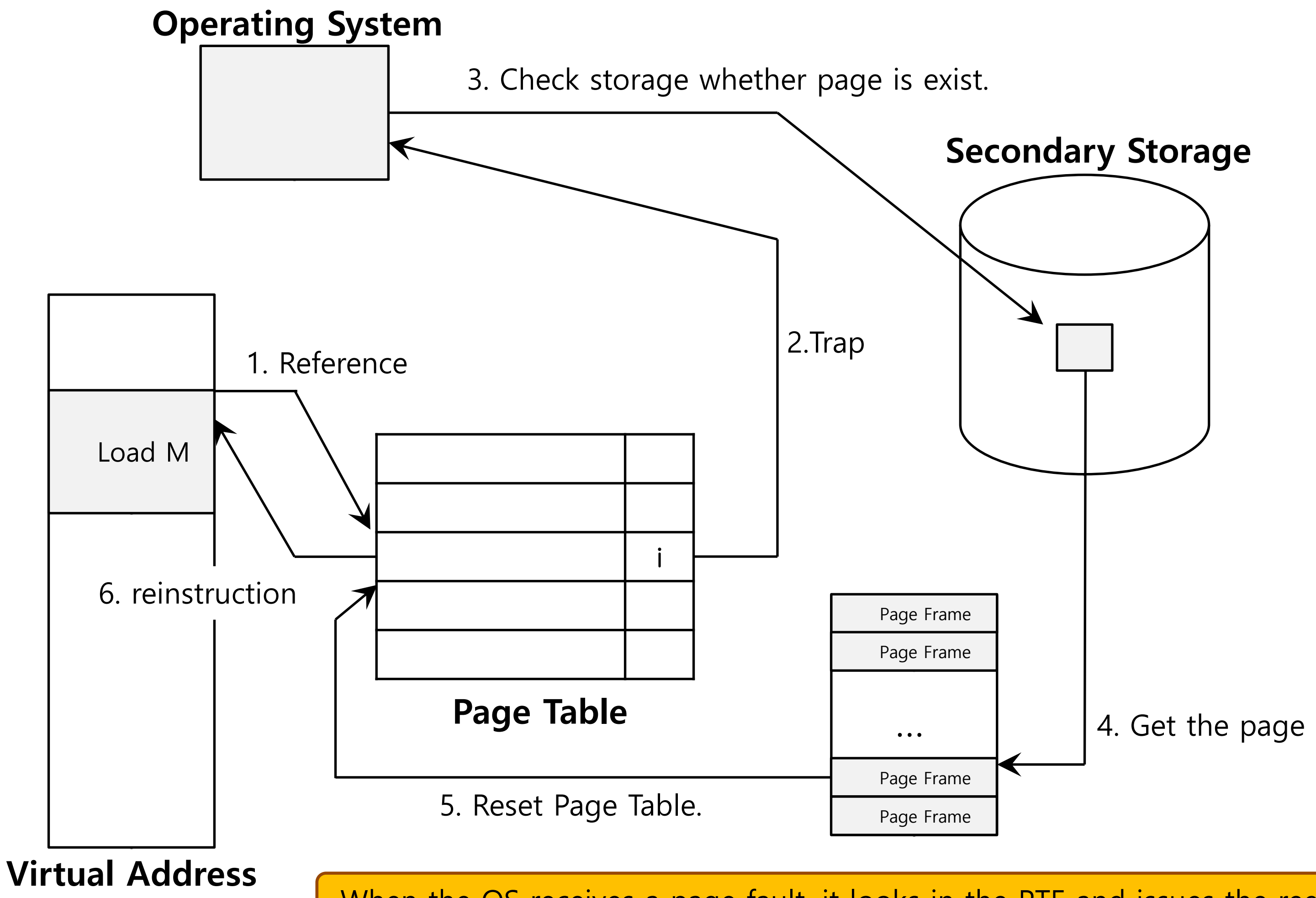
The Page Fault

- 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.

- Confusing terminology sometimes : page-fault also refer to access violations (perhaps page-miss?)

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

```
VPN = (VirtualAddress & VPN_MASK) >> SHIFT
(Success, TlbEntry) = TLB_Lookup(VPN)
if (Success == True) // TLB Hit
if (CanAccess(TlbEntry.ProtectBits) == True)
        Offset = VirtualAddress & OFFSET_MASK
        PhysAddr = (TlbEntry.PFN << SHIFT) | Offset
        Register = AccessMemory(PhysAddr)
else RaiseException(PROTECTION_FAULT)
```

Page Fault Control Flow – Hardware

```
else // TLB Miss
PTEAddr = PTBR + (VPN * sizeof(PTE))
PTE = AccessMemory(PTEAddr)
if (PTE.Valid == False)
        RaiseException(SEGMENTATION_FAULT)
else
if (CanAccess(PTE.ProtectBits) == False)
        RaiseException(PROTECTION_FAULT)
else if (PTE.Present == True)
// assuming hardware-managed TLB
        TLB_Insert(VPN, PTE.PFN, PTE.ProtectBits)
        RetryInstruction()
else if (PTE.Present == False)
        RaiseException(PAGE_FAULT)
```

Page Fault Control Flow – Software

```
PFN = FindFreePhysicalPage()
if (PFN == -1)                      // no free page found
        PFN = EvictPage()           // run replacement algorithm
        DiskRead(PTE.DiskAddr, pfn) // sleep (waiting for I/O)
        PTE.present = True          // update page table with present
        PTE.PFN = PFN               // bit and translation (PFN)
        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 (is a policy not a mechanism).
- Why not via Hardware?

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 or Page *Daemon*: background process that runs in a some sort of "hysteresis"
  - There are fewer than LW pages (low-watermark) available, a background thread that is responsible for freeing memory runs.
  - The thread evicts pages until there are HW pages (high-watermark) available.

- Disclaimer: This lecture slide set has been adapted to AOS course at University of Cantabria by V.Puente. 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 Arpaci-Dusseau (at University of Wisconsin)