Nanyang Polytechnic

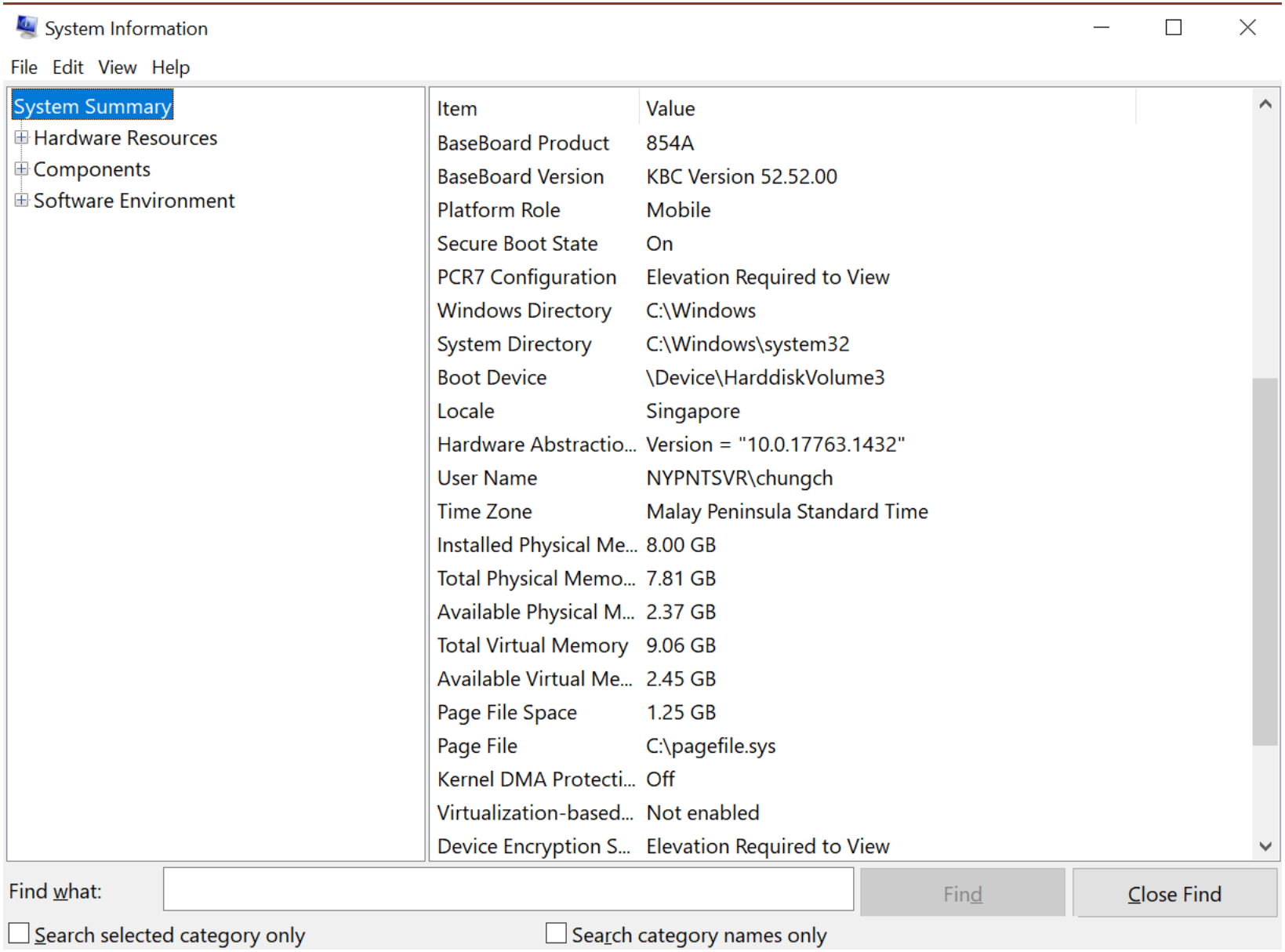
IT2164/IT2561 Operating Systems and Administration

Chapter 9

Virtual Memory

Objectives

- After this lesson, you will be able to :
  - Understand virtual memory, and demand paging
  - Understand virtual memory address mapping to physical memory address


System Information
File Edit View Help
System Summary
Hardware Resources
Components
Software Environment
Item
Value
BaseBoard Product
854A
BaseBoard Version
KBC Version 52.52.00
Platform Role
Mobile
Secure Boot State
On
PCR7 Configuration
Elevation Required to View
Windows Directory
C:\Windows
System Directory
C:\Windows\system32
Boot Device
\Device\HarddiskVolume3
Locale
Singapore
Hardware Abstractio...
Version = "10.0.17763.1432"
User Name
NYPNTSVR\chungch
Time Zone
Malay Peninsula Standard Time
Installed Physical Me...
8.00 GB
Total Physical Memo...
7.81 GB
Available Physical M...
2.37 GB
Total Virtual Memory
9.06 GB
Available Virtual Me...
2.45 GB
Page File Space
1.25 GB
Page File
C:\pagefile.sys
Kernel DMA Protecti...
Off
Virtualization-based...
Not enabled
Device Encryption S...
Elevation Required to View
Find what:
Find
Close Find
Search selected category only
Search category names only

Virtual Memory

- Sole use of primary memory to run programs is not sufficient.
  - Because large portions of most programs are not executed most of the time.
  - Therefore, large portions of programs need not be in memory.
  - Need to maximize the use of primary memory by loading as many programs as possible so as to increase CPU utilization.
- The availability of fast secondary memory (hard disk) allows for its use to supplement primary memory.

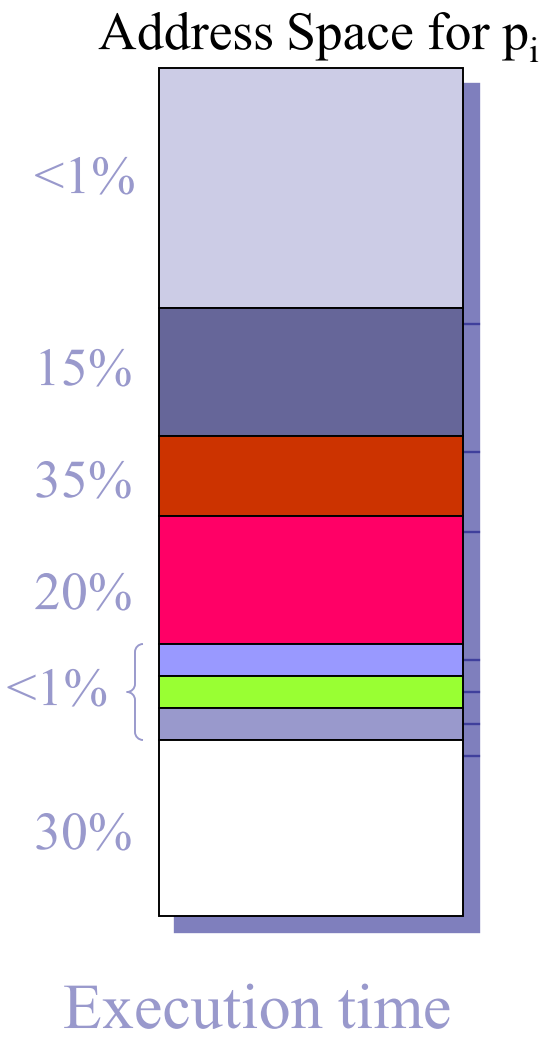
Virtual Memory

- Virtual memory allows the execution of processes that are not completely in memory
- Each process's address space is partitioned into parts that can be loaded into primary memory
- Partitions which are not needed are written to secondary memory
- Allows program that are larger than physical memory to be executed

Locality

- Every process has code and data *locality*
  - Code tends to execute in a few fragments at one time
  - Tend to reference same set of data structures
- As computation moves to a different phase, it changes locality

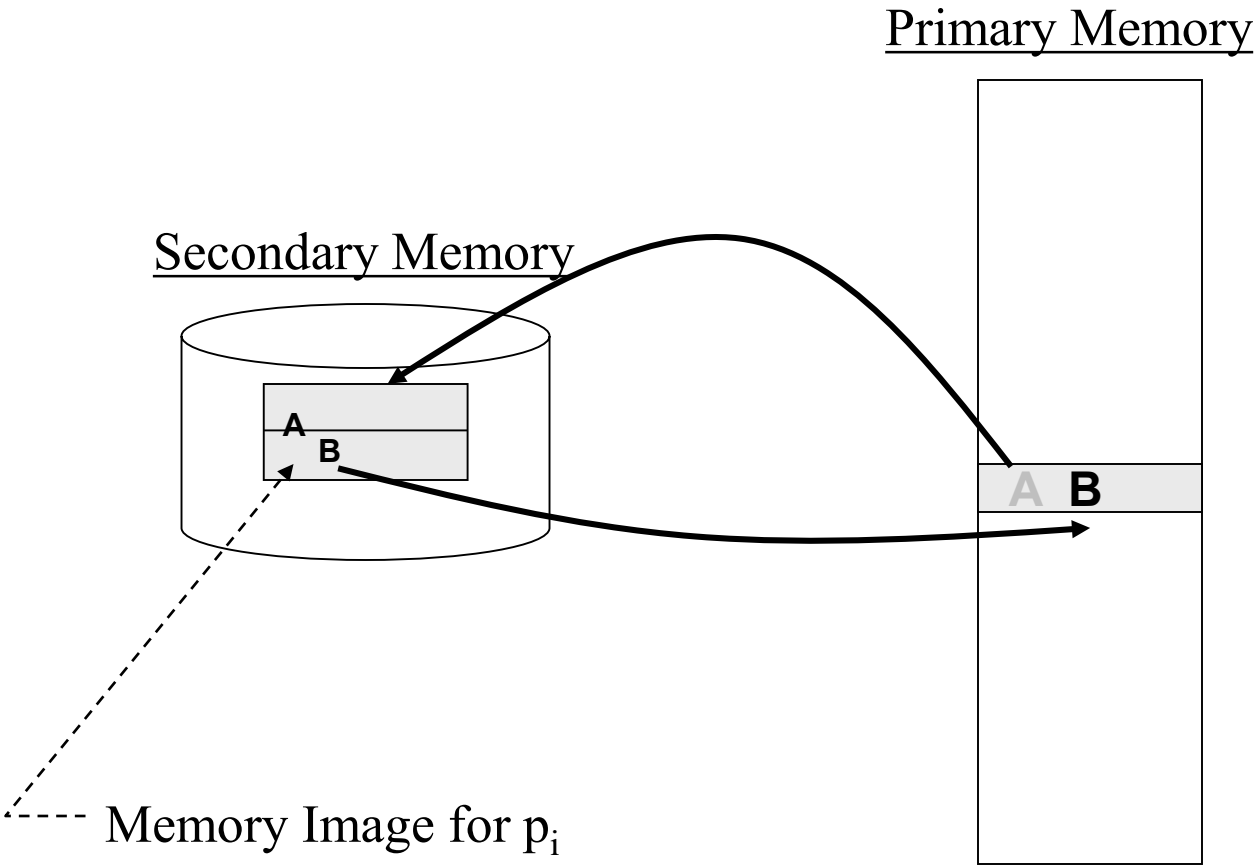
Locality



- Address space is logically partitioned
  - Text, data, stack
  - Initialization, main, error handle
- Different parts have different reference patterns:
  - Initialization code (used once)
  - Code for Φ1
  - Code for Φ2
  - Code for Φ3
  - Code for error 1
  - Code for error 2
  - Code for error 3
  - Data & stack

Φ = phase/stage

Virtual Memory Organization



Virtual Memory: Addresses

- To support virtual memory, we need to implement an addition layer of abstraction.
- Need to implement :
  - Virtual address
    - Every program loaded into memory has a virtual address space.
    - Each space starts from 0x000000.
  - Physical address
    - Actual location in the physical memory.
    - Virtual addresses need to be translated into physical addresses in order to access the contents of that memory location.

Virtual Memory: Size of Blocks of Memory

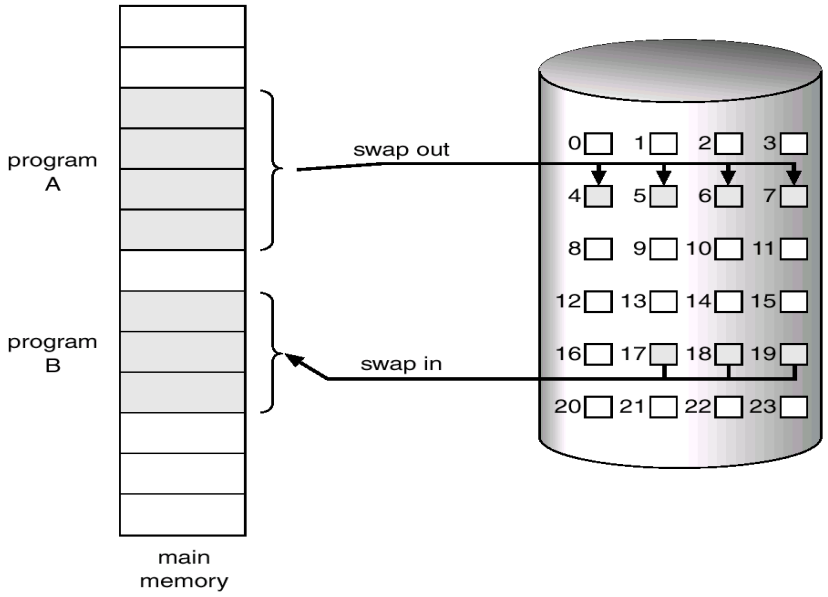
- To support virtual memory, the computer system needs to manage memory in “blocks”.
- Virtual memory system transfers “blocks” of the address space to/from primary memory
- **Fixed size blocks (paging)**:
  - System-defined pages are moved back and forth between primary and secondary memory
- **Variable size blocks (segmentation)**:
  - Programmer-defined segments – corresponding to logical fragments – are the unit of movement
- Demand Paging is the commercially dominant form of virtual memory today

Paging

- A page is a fixed size, 2^h, block of virtual addresses, where h is an integer.
- A page frame is a fixed size, 2^h, block of physical memory (the same size as a page)
- Size of a page is an exponent of 2 to minimize translation time between virtual and physical addresses.

Demand Paging

- In demand paging, the pager brings the necessary pages into memory.
- Avoids reading in memory pages that will not be used, decreasing the swap time and the amount of physical memory needed.
- In a pure demand paging scheme, the system never bring a page into the memory until it is needed.



Benefits

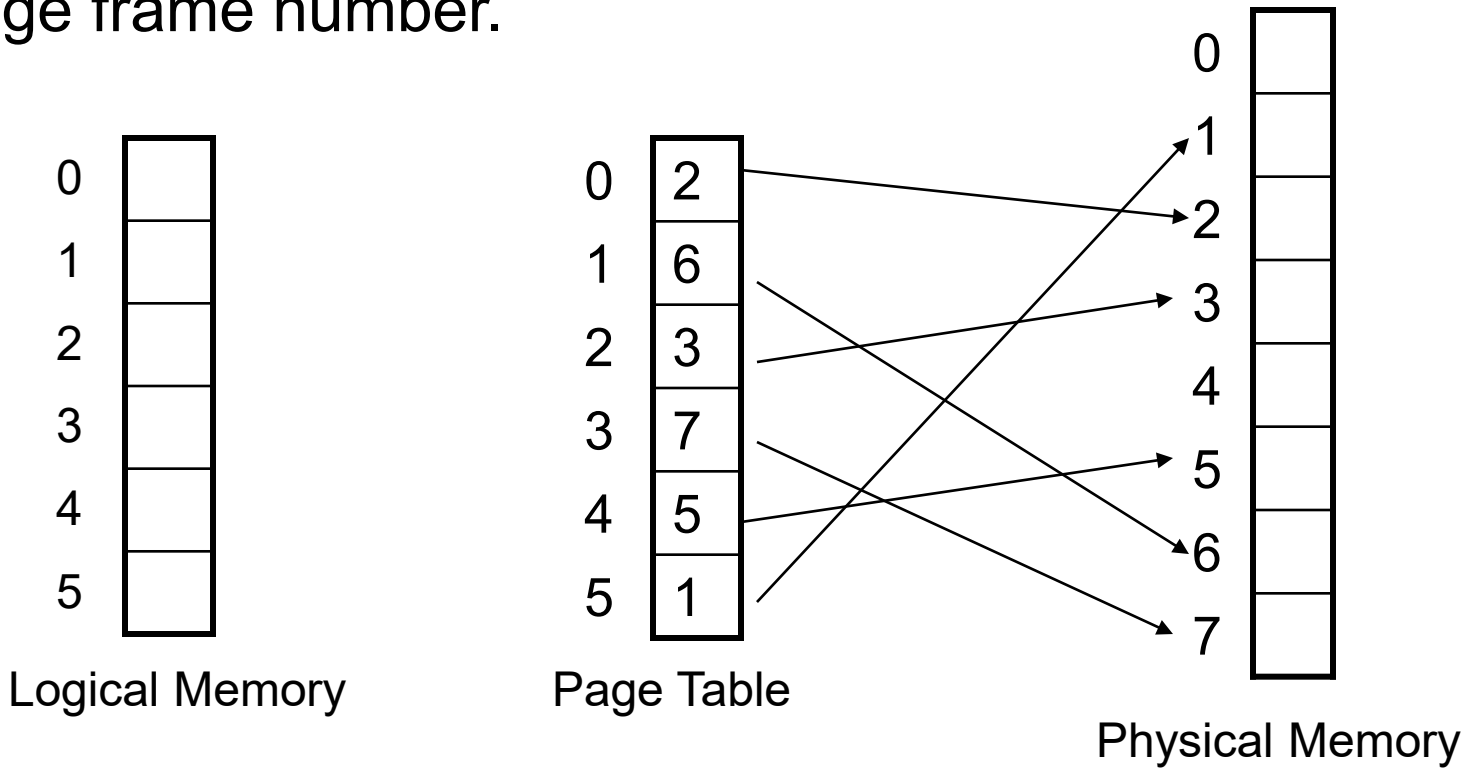
- Programs no longer constrained by the amount of physical memory
- Each program takes less physical memory, allowing more to run at the same time, increasing CPU utilization/throughput

Page Fault

- What happens when a page is required but that page is not in memory but in the virtual memory?
- A page fault occurs.
- A page fault is an event when a page is required but is not found in the primary memory, but is in virtual memory.

Page Table

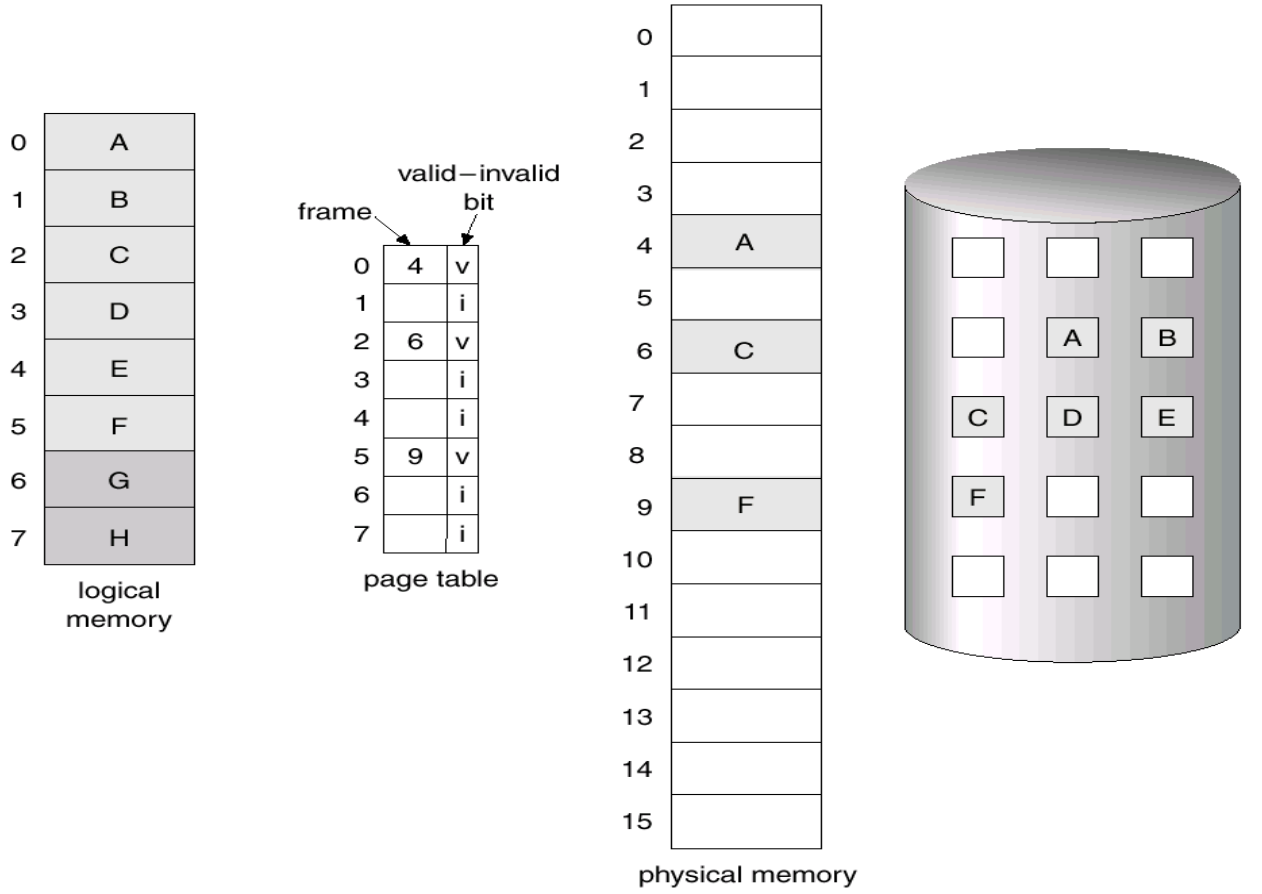
- Page table is a hardware mechanism implemented to map the logical page of the program to the physical page frame where the page is loaded.
- It translates the logical page number to the physical page frame number.



Page Table: Valid-Invalid Bit

- The valid-invalid scheme is used to distinguished between those pages that are in memory and those that are on the disk.
- Each page table entry is associated with a bit (1 - in memory, 0 - not in memory).
- If the bit is set to “valid”, the page is both legal and in memory. Otherwise, the page is either not valid or is valid but is on disk.
- Access to page marked invalid causes a page-fault trap.

Valid-invalid Bit



Performance of Demand Paging

- Page Fault Rate $0 \leq p \leq 1.0$
  - if $p = 0$ no page faults
  - if $p = 1$, every reference is a fault

- Effective Access Time (EAT)

  EAT = (1 – *p*) x memory access time
  + *p* (page fault overhead
  + [swap page out]
  + swap page in
  + restart overhead)

Demand Paging Example

- Memory access time = 1 microsecond
- Average page fault service time is 10 milliseconds or 10,000 microseconds

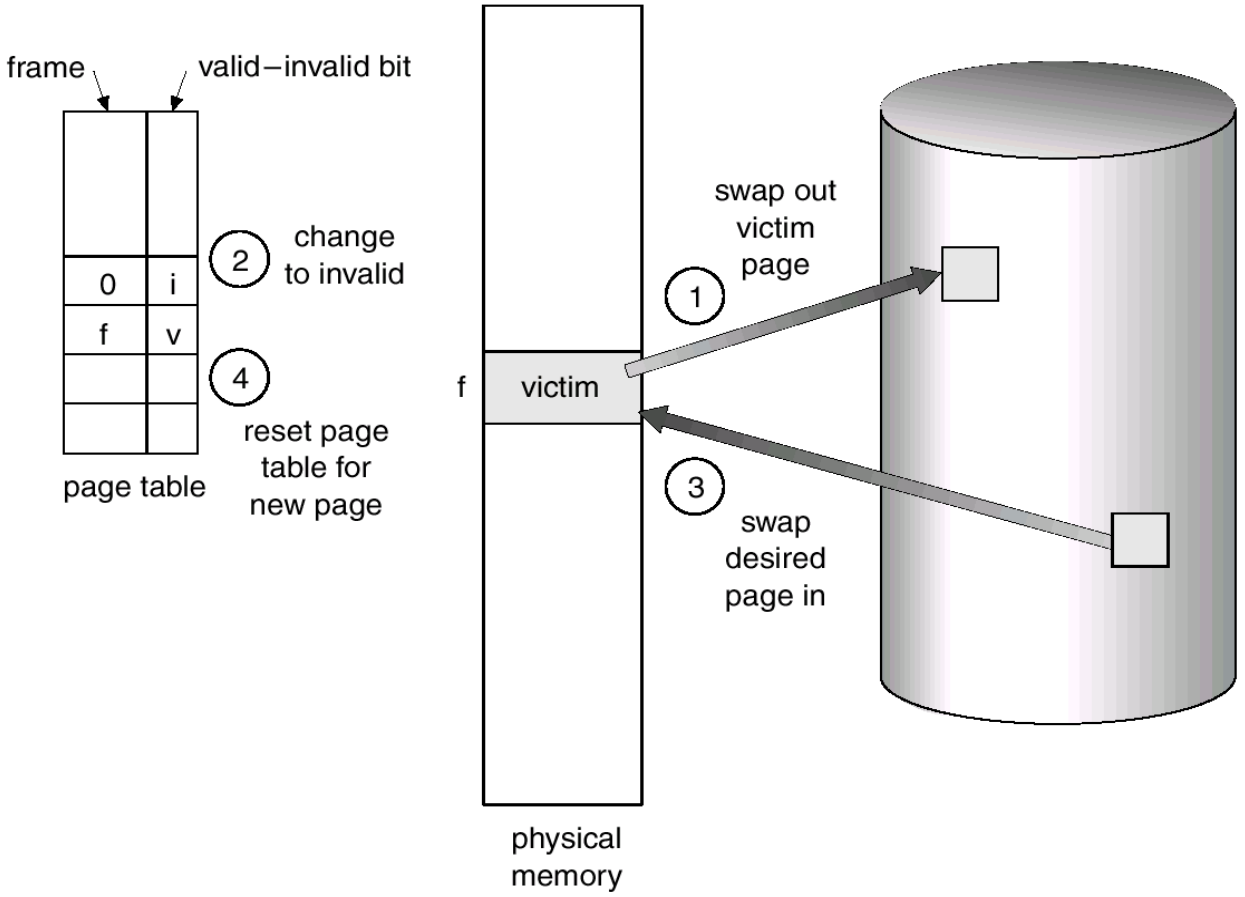
  EAT = (1 – p) x 1 + p x (10000)
  = 1 + 9999p (in microseconds)

- EAT is directly proportional to page-fault rate.

Page Replacement

- Locate the demand page in the disk
- Find a free frame:
  1. If there is a free frame, use it .
  2. Otherwise, use a page replacement algorithm to select a victim frame and write it to the disk. Then change the page and frame tables accordingly.
- Read the desired page into the newly freed frame and change the page and frame tables.
- Restart the user program.
- A modify (dirty) bit is used to reduce overhead of page transfers. Only modified pages are written to disk.

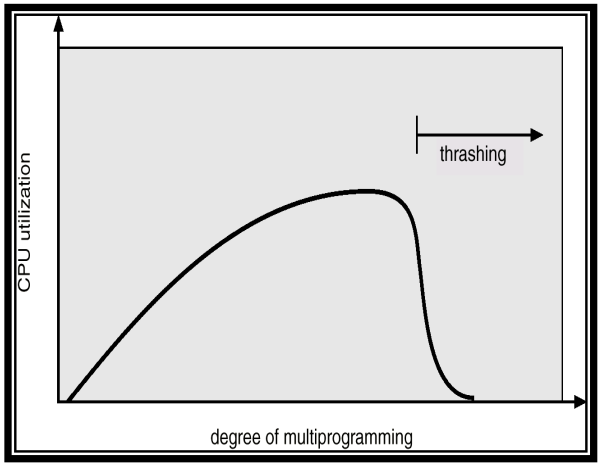
Page Replacement



Thrashing

- If a process does not have “enough” pages, the page fault is very high.
- A process is thrashing if it is spending more time paging than execution.
  - The CPU utilization is low.
  - The operating system thinks that it needs to increase the degree of multiprogramming.
- The effects of thrashing can be limited by using a local(or priority) replacement algorithm.

Thrashing



- Why does paging work?
  - Locality model
    - Process migrates from one locality to another.
    - Localities may overlap.
- Why does thrashing occur?
  - Σ size of locality > total memory size

Conclusion

- Paging is used in most modern OSs.
- The simple design of paging allows for hardware optimizations of the process, allowing it to be implemented successfully in computer systems.
- Virtual memory is essential in maximizing the use of memory in a computer system.