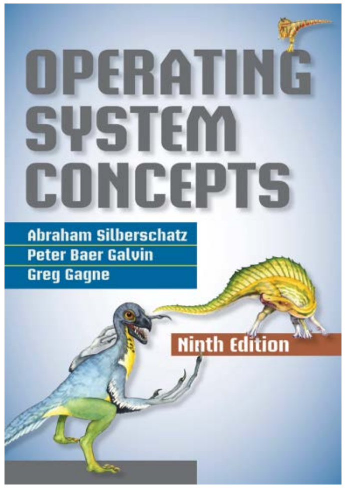
Unit V
Memory Management

Coverage



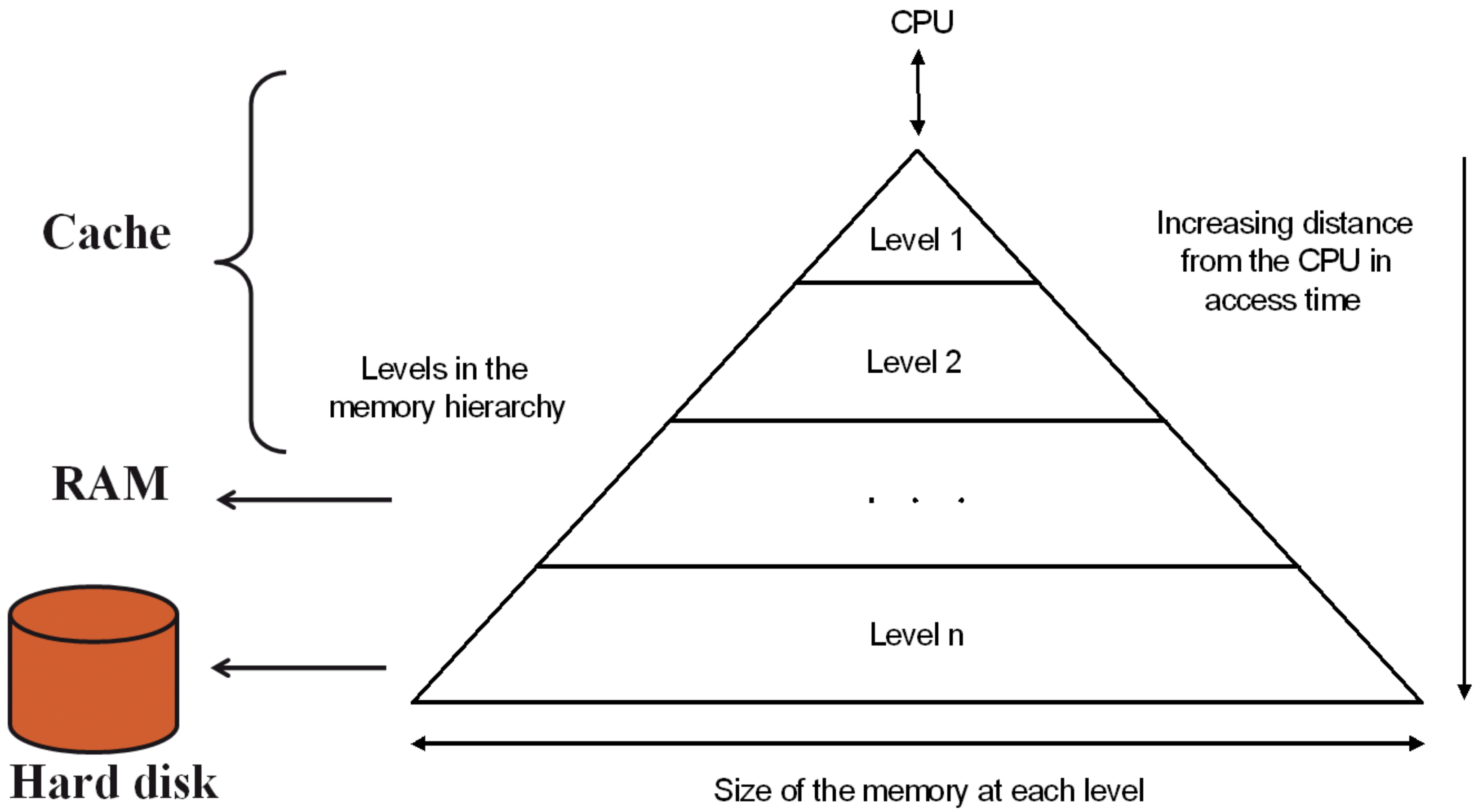
- Basics of memory management
  - Basic Hardware Architecture
  - Hardware address protection
  - Logical Versus Physical Address Space
  - Dynamic Loading
- Swapping
  - Standard Swapping
- Memory Allocation
  - Contiguous Memory Allocation
  - Memory Protection
  - Fixed and Dynamic Partitions
  - Relocatable Dynamic Partitions
  - Fragmentation
- Virtual memory
  - Limitations of Memory Management Algorithms
  - Basic Concept
- Segmentation
  - Basic Method
  - Segmentation Hardware
- Paging
  - Basic Method and Hardware
  - Paging hardware with translation look-aside buffer (TLB)
  - Protection
  - Shared Pages
- Demand Paging
  - Basic Concepts
- Page replacement
  - Basic Page Replacement
  - Algorithms
    - FIFO Page Replacement
    - Optimal FIFO Page Replacement
    - LRU Page Replacement
    - LRU-Approximation Page Replacement
- Allocation of frames
  - Allocation Algorithms
  - Global versus Local Allocation

BASICS OF MEMORY MANAGEMENT

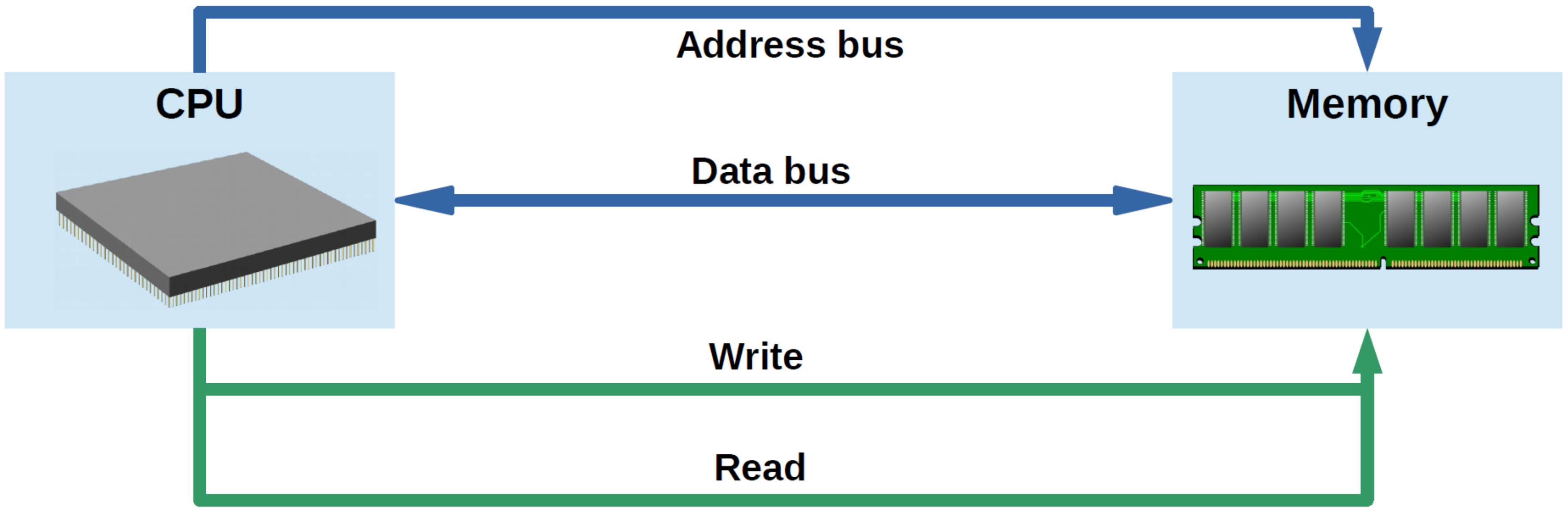
Background

- Program must be brought into memory and placed within a process for it to be run
- Input queue – collection of processes on the disk that are waiting to be brought into memory to run the program

Memory Hierarchy


CPU
Cache
Level 1
Level 2
. . .
Level n
Levels in the memory hierarchy
Increasing distance from the CPU in access time
RAM
Hard disk
Size of the memory at each level

Memory Access



Binding of Instructions and Data to Memory

- Address binding of instructions and data to memory addresses can happen at three different stages
- Compile time: If memory location known a priori, absolute code can be generated; must recompile code if starting location changes
- Load time: Must generate relocatable code if memory location is not known at compile time
- Execution time: Binding delayed until run time if the process can be moved during its execution from one memory segment to another. Need hardware support for address maps (e.g., base and limit registers). A

Logical Versus Physical Address Space

- The concept of a logical address space that is bound to a separate physical address space is central to proper memory management.

- Logical address – generated by the CPU; also referred to as virtual address
- Physical address – address seen by the memory unit □

- Logical and physical addresses are the same in compile-time and load-time address-binding schemes; logical (virtual) and physical addresses differ in execution-time address-binding scheme

Logical and Physical Address

BASIS FOR COMPARISON	LOGICAL ADDRESS	PHYSICAL ADDRESS
Basic	It is the virtual address generated by CPU	The physical address is a location in a memory unit.
Address Space	Set of all logical addresses generated by CPU in reference to a program is referred as Logical Address Space.	Set of all physical addresses mapped to the corresponding logical addresses is referred as Physical Address.
Visibility	The user can view the logical address of a program.	The user can never view physical address of program
Access	The user uses the logical address to access the physical address.	The user can not directly access physical address.
Generation	The Logical Address is generated by the CPU	Physical Address is Computed by MMU

Dynamic Loading

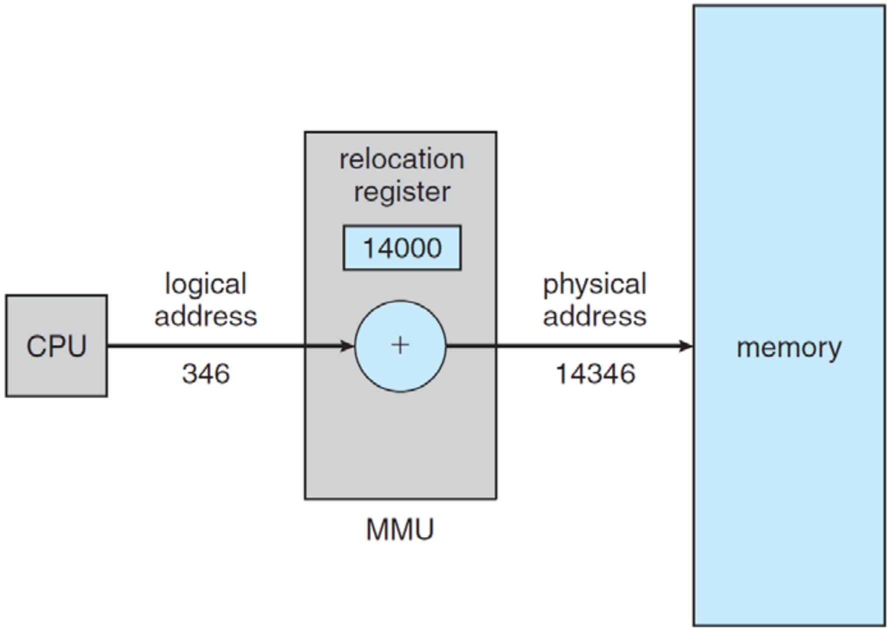


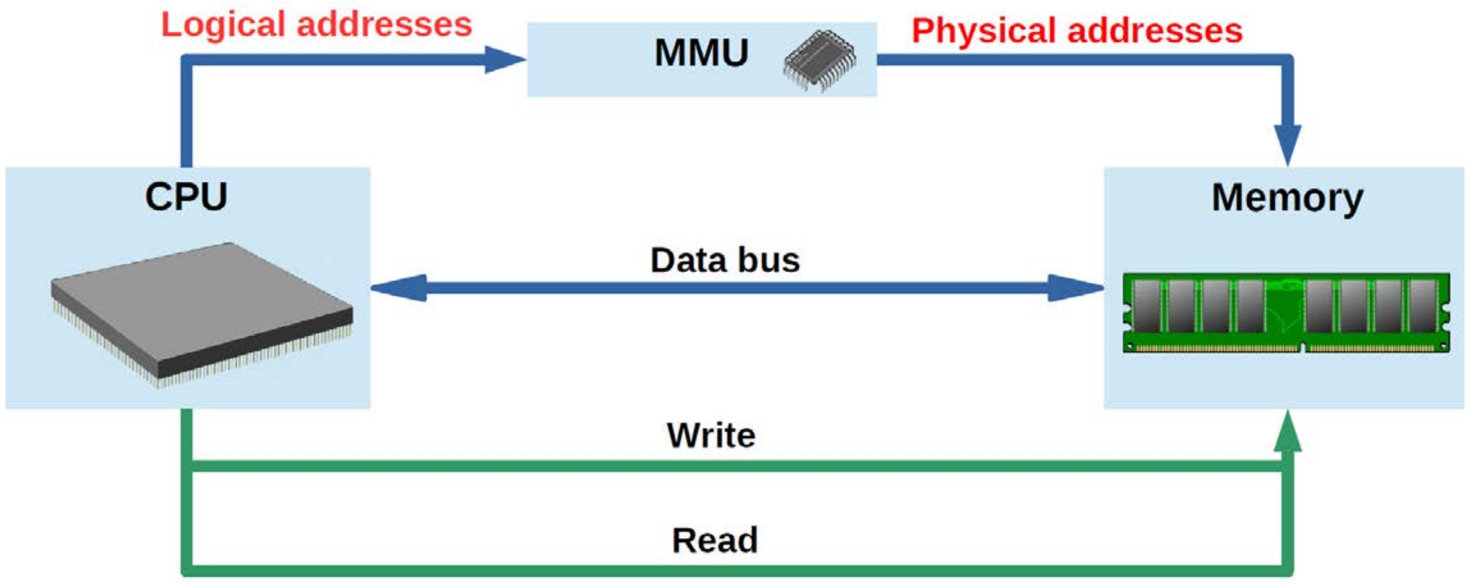
Figure 8.4 Dynamic relocation using a relocation register.

Using a relocation register (as a base register)

- **Added to every logical address generated by a user process at run time**
- **Make sure each process has a separate memory space**

Memory-Management Unit (MMU)

- Hardware device that maps virtual to physical address
- In MMU scheme, the value in the relocation register is added to every address generated by a user process at the time it is sent to memory
- The user program deals with logical addresses; it never sees the real physical addresses



Basic Hardware Architecture

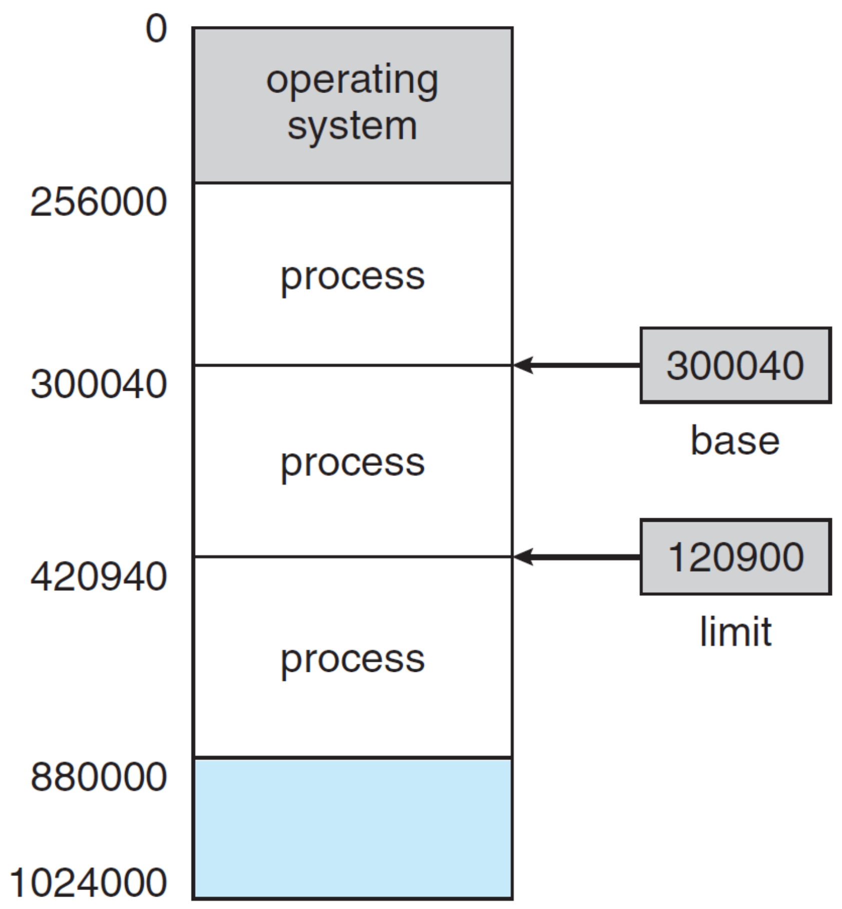


Figure 8.1 A base and a limit register define a logical address space.

Hardware address protection

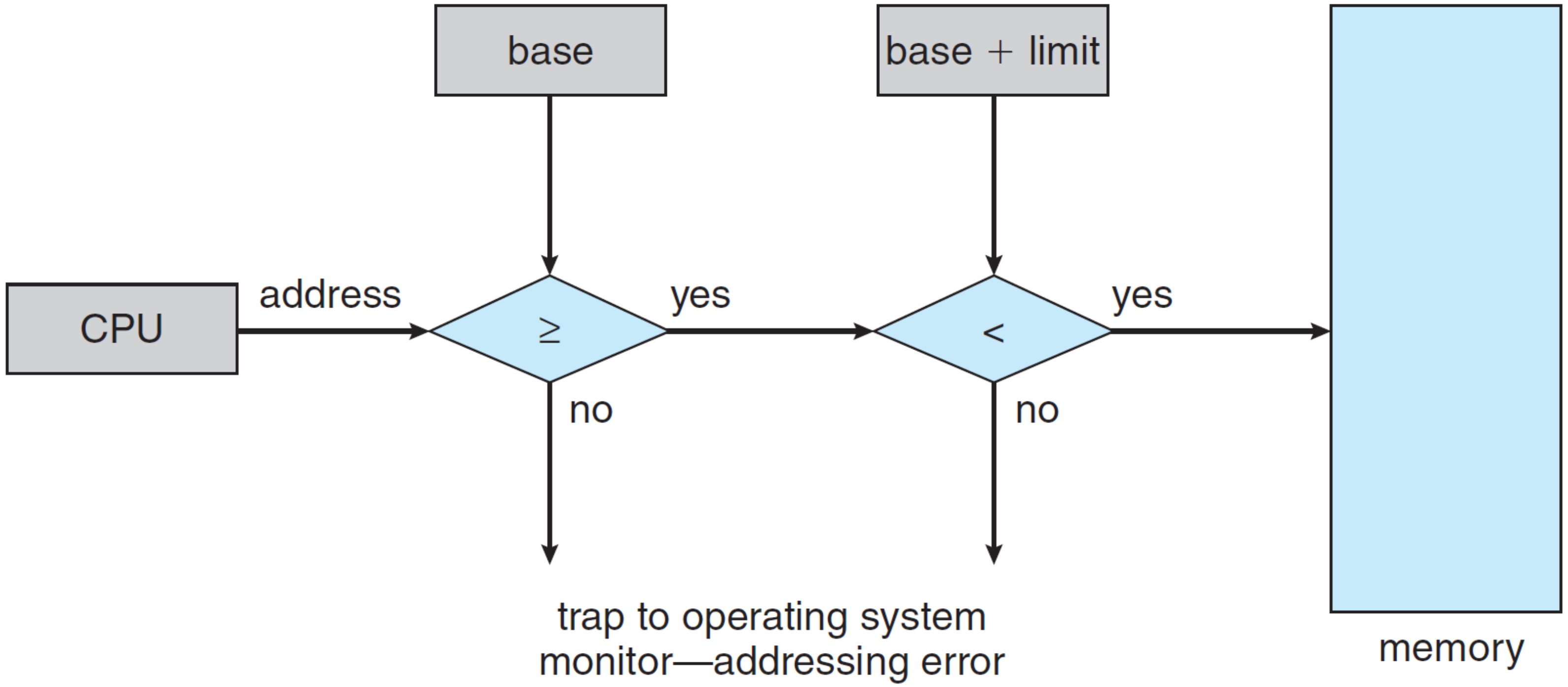


Figure 8.2 Hardware address protection with base and limit registers.

Dynamic Loading

- Static Loading
  - Loading can be done in the beginning before the actual execution.
- Dynamic Loading
  - A routine is not loaded until, it is called.
- **Dynamic loading** mechanism, at run time, **loads** a **library** into memory, retrieve the addresses of **functions** and variables contained in the **library**, execute those **functions** or access those variables, and unload the **library** from memory.
- Better memory-space utilization; unused routine is never loaded
- Useful when large amounts of code are needed to handle infrequently occurring cases □
- No special support from the operating system is required implemented through program design

Swapping

- A process can be swapped temporarily out of memory to a backing store, and then brought back into memory for continued execution

- Backing store – a fast disk large enough to accommodate copies of all memory images for all users; must provide direct access to these memory images

- Roll out, roll in – swapping variant used for priority-based scheduling algorithms; lower-priority process is swapped out so higher-priority process can be loaded and executed

- Major part of swap time is transfer time; total transfer time is directly proportional to the amount of memory swapped

Dynamic Linking

- Static linking vs. Dynamic linking
- Linking postponed until execution time □
- Small piece of code, stub, used to locate the appropriate memory-resident library routine □
- Stub replaces itself with the address of the routine, and executes the routine
- Operating system needed to check if routine is in processes' memory address
- Dynamic linking is particularly useful for libraries

Schematic View of Swapping

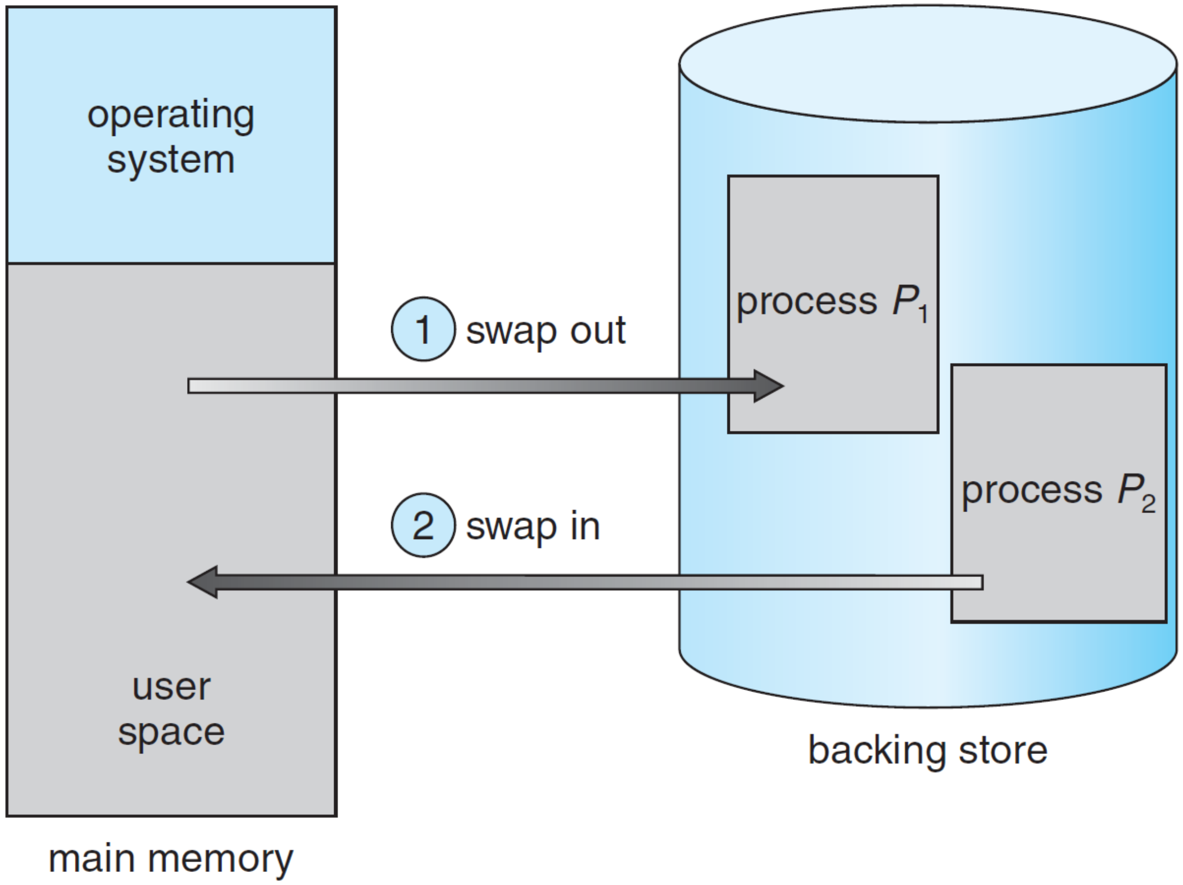


Figure 8.5 Swapping of two processes using a disk as a backing store.

When to Swap?

- **Swapping is expensive**!
  - How much time at least is necessary to swap in a 1 GB process (e.g., Firefox) with a transfer rate of 500 MB / second (Typical transfer rate for a (very good) SSD)
  - May have to perform a swap out before!

Swap out

- When?
  - Memory occupied over threshold
  - A memory allocation request fails
- Which process to swap out?
  - Recently executed processes (RR scheduling)
  - Processes with lower priorities (Priority-based scheduling)

Swap in

- When a process is ready to execute
  - I/O completed
- When a large amount of memory freed

MEMORY ALLOCATION

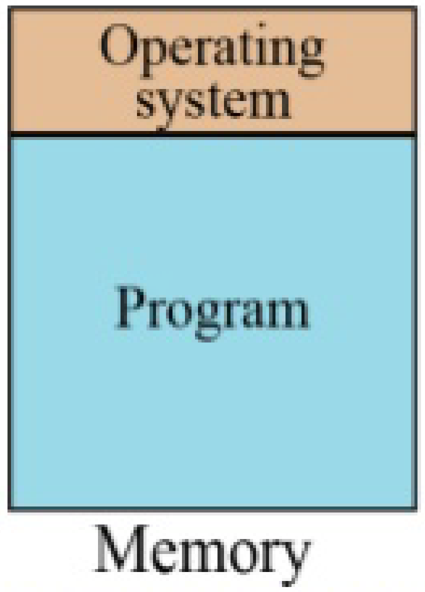


FIG 5(a): mono programming

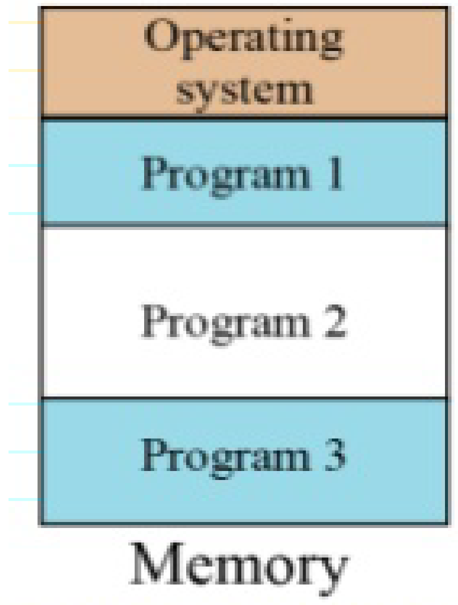


FIG 5(b): multiprogramming

Memory Allocation

- Main issues
  - Keep track of memory allocations
  - Return requested memory chunks as fast as possible
  - Avoid fragmentation

- Allocation unit
  - Smallest amount of continuous memory managed by the OS
  - From a few bytes to several KBytes
  - Impact of this size?

- Keeping track of allocation units
  - Bitmaps
  - Linked Lists

Multiprogramming: Issues

- Process Permission
  - OS estimates the required memory and allocates it
- Dynamic Allocation
  - A process may request additional memory space
  - A process may release part of its memory space
- Process termination
  - OS must release all the allocated memory

Contiguous Memory Allocation

- The main memory must accommodate both the operating system and the various user processes.

- The memory is usually divided into two partitions, one for the resident operating system, and one for the user processes.

- It is possible to place the operating system in either low memory or high memory.

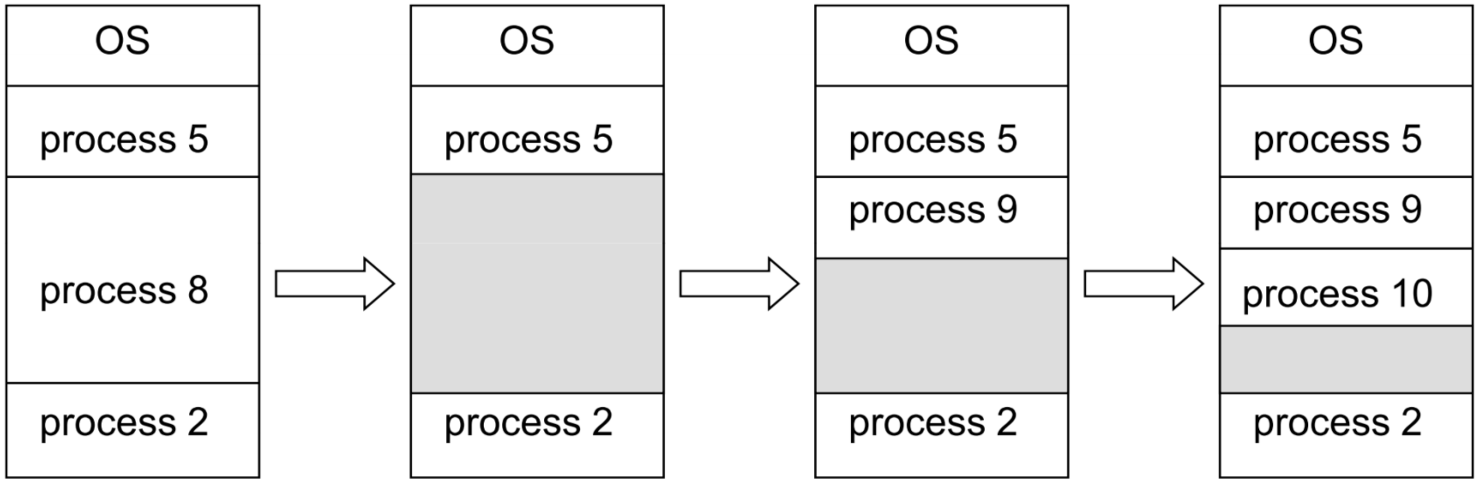
- fixed-partition allocation
- fitted-partition allocation

Fixed-partition allocation

- Memory is divided into several fixed-sized partitions
- Relocation-register scheme used to protect user processes from each other, and from changing operating-system code and data
- Relocation register contains value of smallest physical address; limit register contains range of logical addresses – each logical address must be less than the limit register

Fitted-partition allocation

- Hole – block of available memory; holes of various size are scattered throughout memory
- When a process arrives, it is allocated memory from a hole large enough to accommodate it
- Operating system maintains information about:
- a) allocated partitions b) free partitions (hole)



Bitmaps vs. Linked Lists

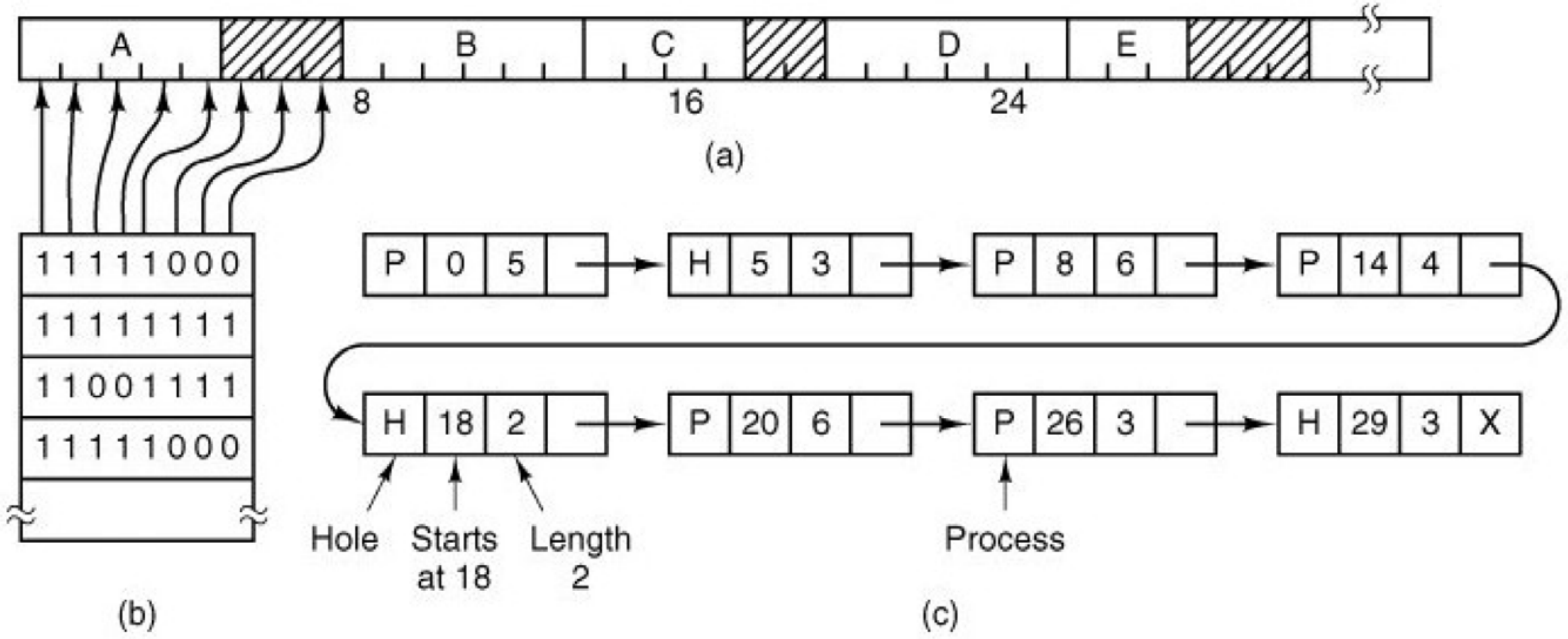


Fig:(a) A part of memory with five processes and three holes. The tick marks show the memory allocation units. The shaded regions (0 in the bitmap) are free. (b) The corresponding bitmap. (c) The same information as a list.

HW address protection with base and limit registers

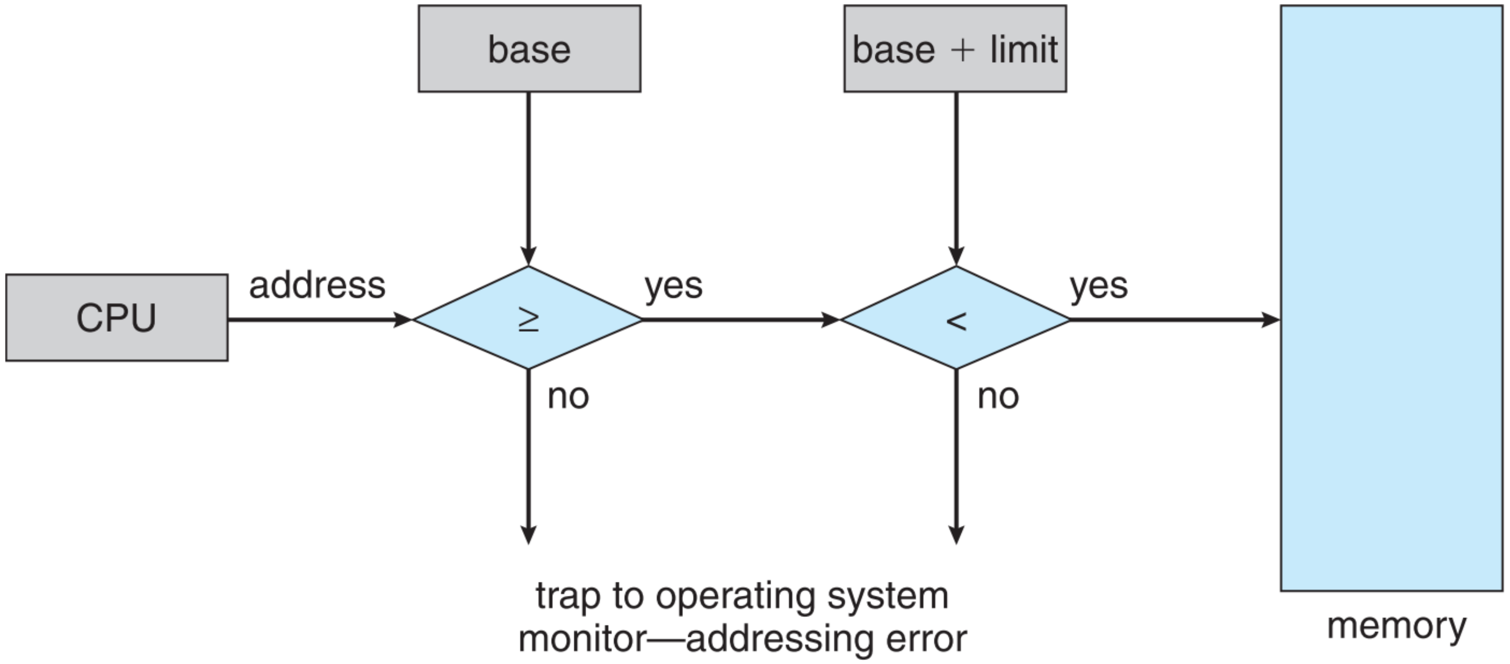


Figure 8.2 Hardware address protection with base and limit registers.

Dynamic Storage -Allocation Problem

- How to satisfy the request of size n from a list of free holes

- **First Fit**
  - The system **scans the list** along **until** a large enough continuous allocation is found

- **Next Fit**
  - Scanning begins at the last position where a free block has been found
  - Performs slightly worse than First Fit

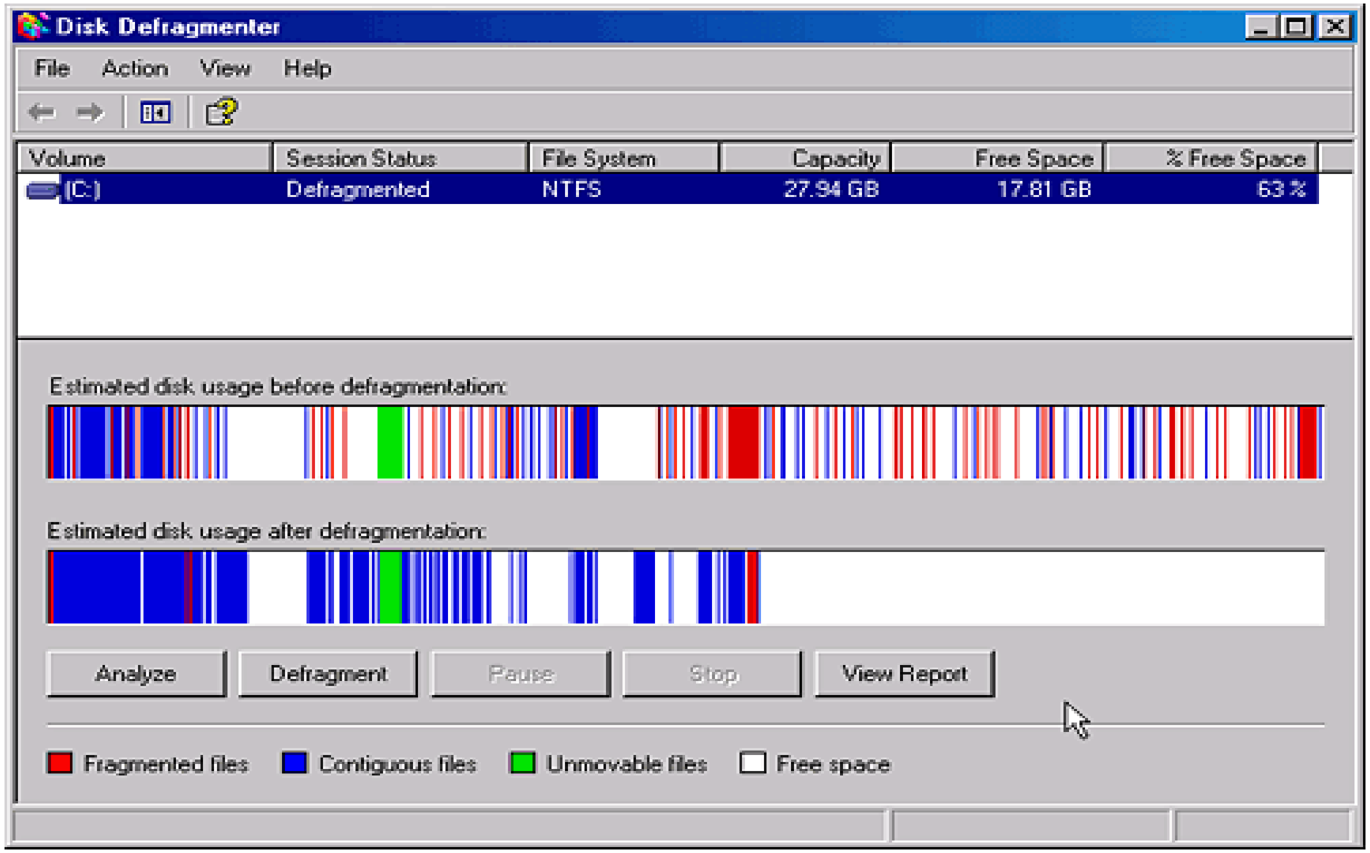
- **Best Fit**
  - **Scans all the list** and takes the smallest free block that is adequate
  - Performs worse than First Fit !
    - Can you guess why?

Allocation Algorithms (Cont.)

- **Worse Fit**
  - Searches for the largest free block
  - Not very good either!

- **Quick Fit**
  - Separate lists for most usual size
  - Additional complexity of memory management
    - Merging is expensive
  - But very quick search!

Fragmentation



Volume	Session Status	File System	Capacity	Free Space	% Free Space
(C:)	Defragmented	NTFS	27.94 GB	17.81 GB	63 %

Estimated disk usage before defragmentation:

Estimated disk usage after defragmentation:

🟥 Fragmented files 🟦 Contiguous files 🟩 Unmovable files ⬜ Free space

Fragmentation

- External Fragmentation – free memory space is broken into little pieces
  - there is enough total memory space to satisfy a request, but it is not contiguous, unable to serve the request
- Internal Fragmentation – allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used
  - break memory into fixed-sized blocks and allocate memory in units based on block size
  - reduce the number of small external fragmentation
- Reduce external fragmentation by compaction
  - Shuffle memory contents to place all free memory together in one large block
  - Compaction is possible only if relocation is dynamic, and is done at execution time
  - I/O problem: I/O operations usually use physical addresses
    - Latch job in memory while it is involved in I/O
    - Do I/O only into OS buffers

MAIN MECHANISMS

Main Mechanisms

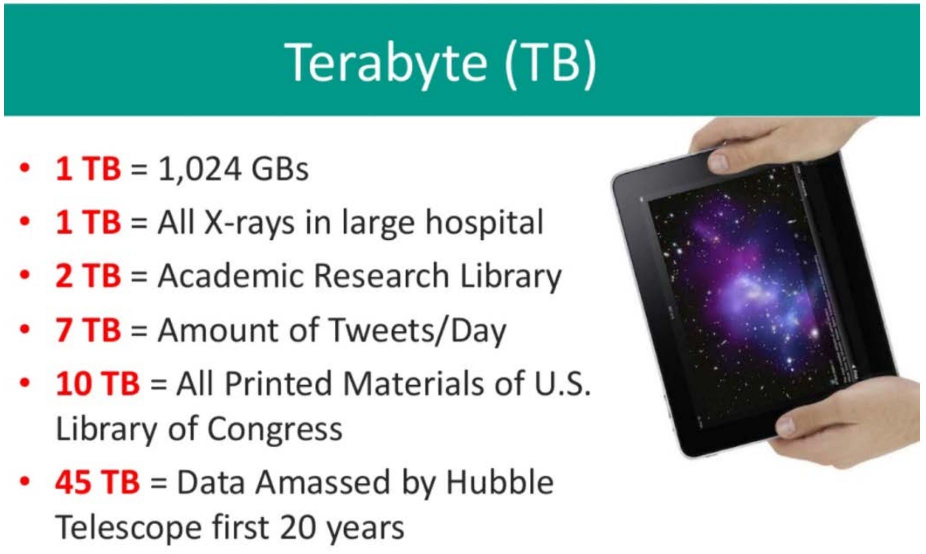
- Swapping
- Virtual Memory
- Segmentation and Paging

Virtual Memory

- Virtual Memory is a memory that appears to exist as main storage although most of it is supported by data held in secondary storage, transfer between the two being made automatically as required.
- If the size of the program is greater than the available memory size, then the concept of virtual memory is used.
- Ever wondered how a 10GB Game like God Of War Cry fits into your 2GB RAM computer?
  - Can you guess?

Address space of Windows

- 32-bit Address Space
  - 32-bits = 2^{32} = 4 GB
  - 3 GB for address space
  - 1 GB for kernel mode



Terabyte (TB)

- **1 TB** = 1,024 GBs
- **1 TB** = All X-rays in large hospital
- **2 TB** = Academic Research Library
- **7 TB** = Amount of Tweets/Day
- **10 TB** = All Printed Materials of U.S. Library of Congress
- **45 TB** = Data Amassed by Hubble Telescope first 20 years

- 64-bit Address space
  - 64-bits = 2^{64} = 7,179,869,184 GB=17,179,869 TB
  - x64 supports 48 bits virtual = 262,144 GB = 256 TB
  - IA-64 support 50 bits virtual = 1,048,576 GB = 1024 TB
  - 64-bit Windows supports 44 bits = 16,384 GB = 16 TB

Advantages of VM

- Translation
  - Program can be given consistent view of memory, even though physical memory is scrambled
  - Only the most important part of program must be in Physical memory.
  - Contiguous structures (like stacks) use only as much physical memory as necessary yet grow later.
- Protection
  - Different threads (or processes) protected from each other
  - Different pages can be given special behavior
    - (Read Only, Invisible to user programs, etc).
  - Kernel data protected from User programs
  - Very important for protection from malicious programs
    - => Far more "viruses" under Microsoft Windows
- Sharing
  - Can map same physical page to multiple users ("Shared memory")

Disadvantages of VM

- Applications run slower if the system is using virtual memory
- It Takes more time to switch between applications
- Less hard drive space for your use
- Less hard drive space for your use
- It reduces system stability

Segmentation

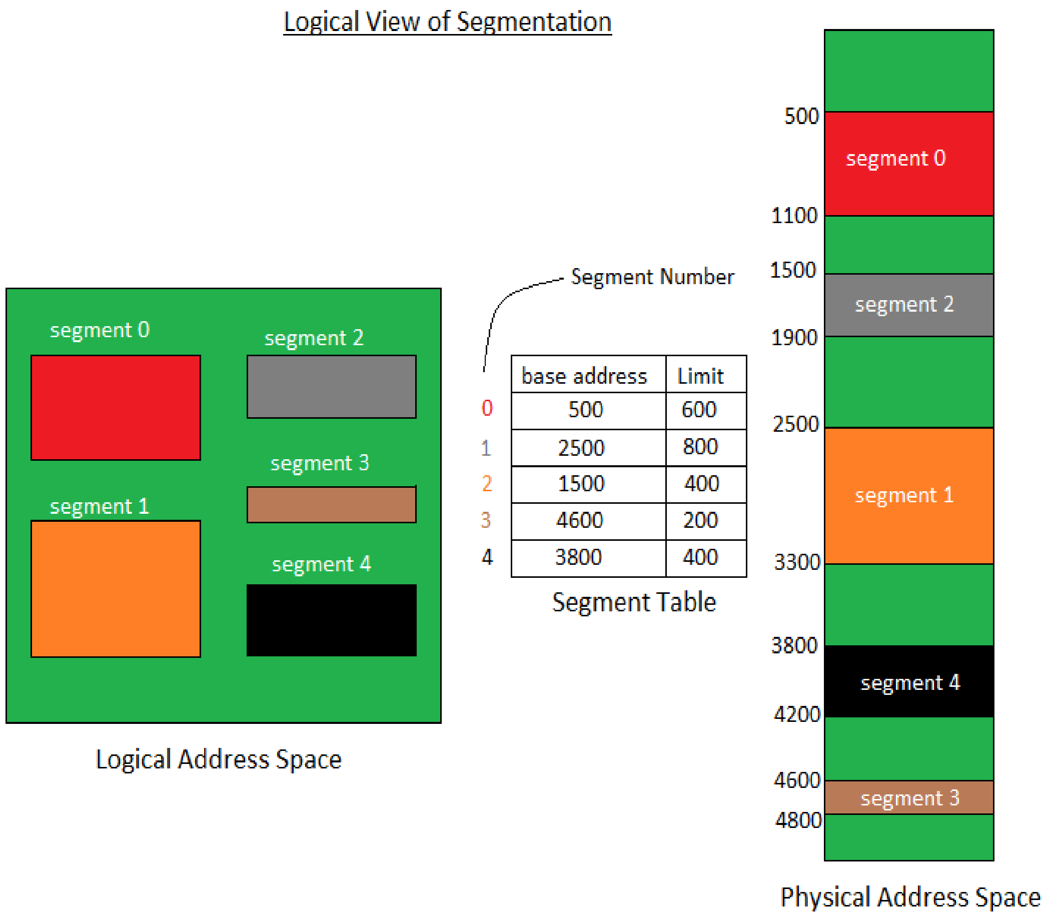
- A Memory Management technique in which memory is divided into variable sized chunks which can be allocated to processes. Each chunk is called a **Segment**.
- A table stores the information about all such segments and is called **Segment Table.**

- **Segment Table:** It maps two dimensional Logical address into one dimensional Physical address.
- It's each table entry has
  - **Base Address:** It contains the starting physical address where the segments reside in memory.
  - **Limit:** It specifies the length of the segment.

Logical View of Segmentation

Translation of Two dimensional Logical Address to one dimensional Physical Address.

- Address generated by the CPU is divided into:
  - Segment number**(s):** Number of bits required to represent the segment.
  - Segment offset **(d)**: Number of bits required to represent the size of the segment**.**



	base address	Limit
0	500	600
1	2500	800
2	1500	400
3	4600	200
4	3800	400

Segmentation

Advantages of Segmentation:

- No Internal fragmentation.
- Segment Table consumes less space in comparison to Page table in paging.

Disadvantage of Segmentation:

- As processes are loaded and removed from the memory, the free memory space is broken into little pieces, causing External fragmentation.

Limitations of Segmentation

- Fragmentation
  - Solution: Using algorithms to select segments (e.g., best-fit, first-fit, etc.)
- Cost of segment expansion!
  - If a process allocates more space in a segment and this segment cannot be expanded, and there is free memory available elsewhere

  → **memory segment must be moved**
  - 1. Process is blocked
  - 2. OS makes a memory copy ! segment is moved to another location
  - 3. Process is unblocked
- ⇒ Paging!

Basics of Paging

- Paging allows the logical address space of a process to be non contiguous in physical memory
- Physical memory
  - All physical memory is cut into fixed-size blocks
  - Physical memory includes swap partitions
  - Logical memory: **page**
  - Physical memory: **frame**
- Virtual memory
  - Address divided into two parts
    - Page number (p)
    - Page offset (d)

Paging

- The most common kind of automatic virtual memory is called paging.
- In paging, the program is cut up arbitrarily into pieces. Each piece is called a page and contains the same number of bytes as every other page--say, 512.
- There are many more pages than will fit into main memory at once, so most of them stay on the disk.
- The processor knows only about byte addresses in one large address space called the *virtual address space*. Every time the processor accesses a byte, the address of the byte is checked.
- The high-order bits of the address tell which page contains that byte. (The low-order bits tell which byte within the page.)
- If that page is not in main memory, the user program stops. The virtual memory program starts up, finds an old page, moves it to the disk, and brings the desired page into memory in its place. (We will use the term "memory" to refer to the fast, main memory only.)
- The act of discovering that a page is needed from the disk and bringing it into memory is called a *page fault.*

Basic Hardware

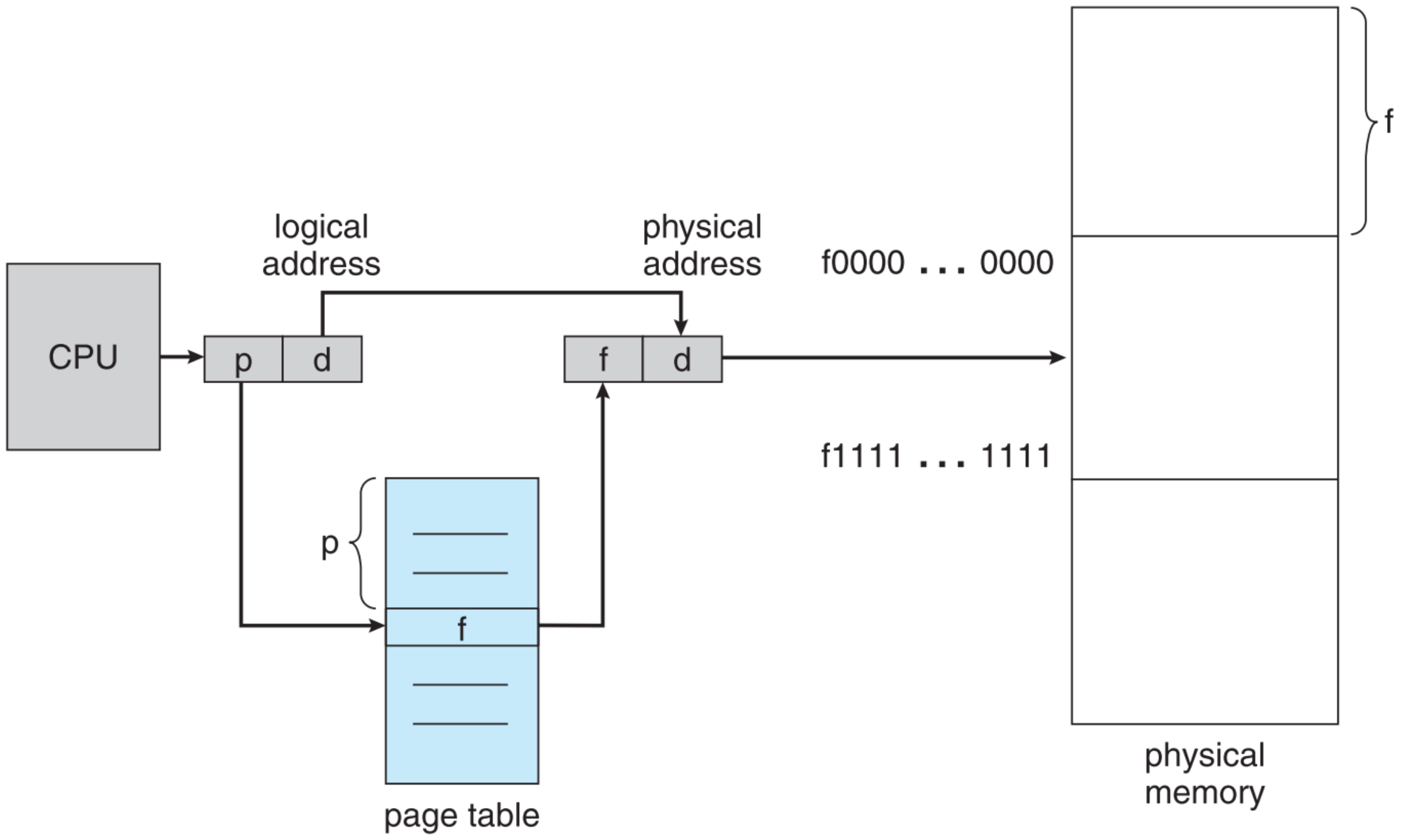
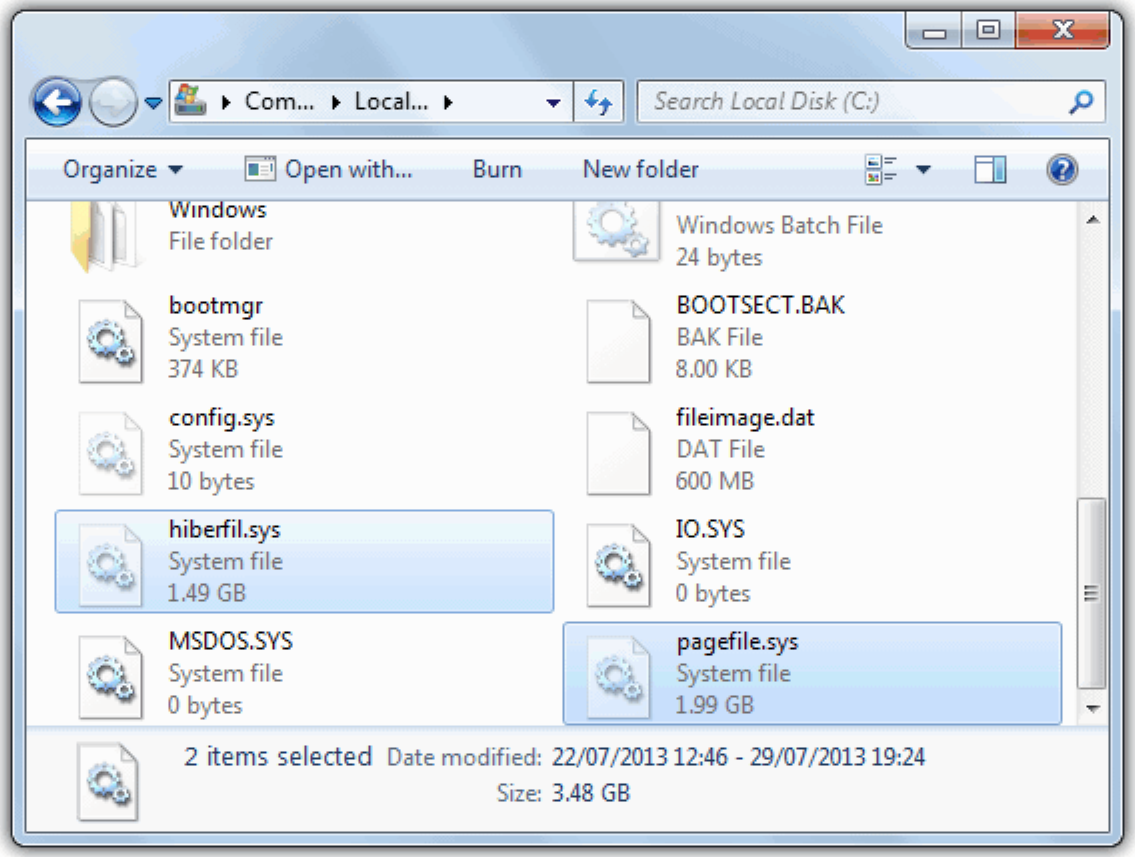


Figure 8.10 Paging hardware.

Paging

In Window OS

- The page file is a hidden file called pagefile.sys
- It is regenerated at each boot .
- There is no need to include it in a backup



Windows
File folder

Windows Batch File
24 bytes

bootmgr
System file
374 KB

BOOTSECT.BAK
BAK File
8.00 KB

config.sys
System file
10 bytes

fileimage.dat
DAT File
600 MB

hiberfil.sys
System file
1.49 GB

IO.SYS
System file
0 bytes

MSDOS.SYS
System file
0 bytes

pagefile.sys
System file
1.99 GB

2 items selected Date modified: 22/07/2013 12:46 - 29/07/2013 19:24
Size: 3.48 GB

Demand Paging

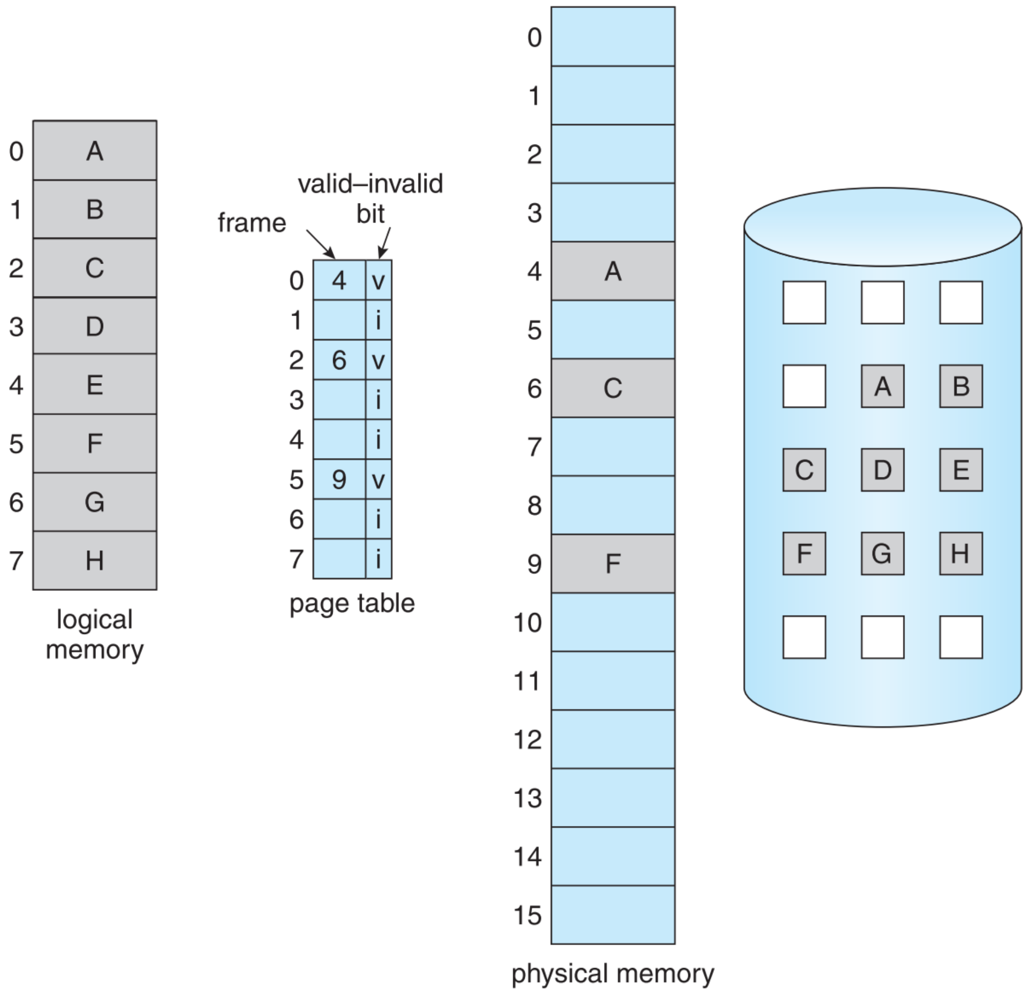


Figure 9.5 Page table when some pages are not in main memory.

Advantages

- An advantage of paging is that it works regardless of the contents of the pages.
- The mechanism needed to determine whether a given page is in memory is simple.
- Many computers have special hardware to speed up the translation between an address in the virtual space and a page in memory.

Disadvantages

- If the program needs a particular byte, the entire page surrounding that byte must be brought into memory.
- If no other bytes on that page are useful at the moment, a large amount of main memory is wasted.
- Since programs are cut up arbitrarily into pages in the first place, it is common that the rest of the page has nothing to do with the part currently wanted.
- Sometimes a significant fraction of memory is taken up by pages from which the processor wants only a few bytes.
- These pages crowd out pages containing other parts of the program, causing many pages to be swapped to run the rest of the program.
- The many accesses to slow secondary memory cause the whole system to be slow.

Page Replacement Algorithms

- In a computer operating system that uses paging for virtual memory management, page replacement algorithm decide which memory pages to page out.
- When a page of memory need to be allocated
  - FIFO
  - LRU
  - OPT

FIFO

First in first out is very easy to implement.
The fifo algorithm select the page for replacement
that has been in memory the longest time.

time	1	2	3	4	5	6	7	8	9	10	11	12
page	P2	p3	p2	p1	p5	p2	p4	p5	p3	p2	p5	p2
	p2*	p2*	p2*	p2*	P5	p5	p5*	p5*	P3	P3	P3	P3*
	🟦	p3	p3	P3	p3*	P2	P2	P2	p2*	P2*	P5	P5
	🟦	🟦	🟦	P1	p1	P1*	p4	p4	p4	p4	P4*	p2
			hit					hit		hit		

LRU

- The least recently used page replacement algorithm keeps track page uses over a short period of time.
- The LRU algorithm can be implemented by associating a counter with every page that is n main memory.

time	1	2	3	4	5	6	7	8	9	10	11	12
pag e	P2	p3	p2	p1	p5	p2	p4	p5	p3	p2	p5	p2
	p2*	p2*	P2	P2	p2*	P2	P2	p2*	P3	P3	P3*	P3*
	🟦	p3	p3*	p3*	P5	P5	p5*	P5	P5	P5*	P5	P5
	🟦	🟦	🟦	P1	P1	P1*	p4	P4	p4*	P2	P2	p2
			hit			hit		hit			hit	hit

OPT

- The optimal policy selects that page for replacement for which the time to the next reference is longest.
- This algorithm result is fewest number of page faults.

time	1	2	3	4	5	6	7	8	9	10	11	12
page	P2	p3	p2	p1	p5	p2	p4	p5	p3	p2	p5	p2
	p2	p2	P2	P2	P2	P2	P4	P4	P4	P2	P2	P2
	🟦	p3	p3	p3	P3	P3	P3	P3	P3	P3	P3	P3
	🟦	🟦	🟦	P1	P5	P5	p5	P5	p5	P5	P5	P5
			hit			hit		hit	hit		hit	hit

Disadvantage of OPT

- It is impossible to have knowledge about future references
- Try approximations that use the past to predict the future. For example
  - a page used recently is likely to be used in the near future
  - a page unused recently is unlikely to be used in the near future

Allocation of frames

- A process should allocate enough frames for its current locality.
- If more frames are allocated, there is little improvement.
- If less frames are allocated, there is a very high page fault rate.

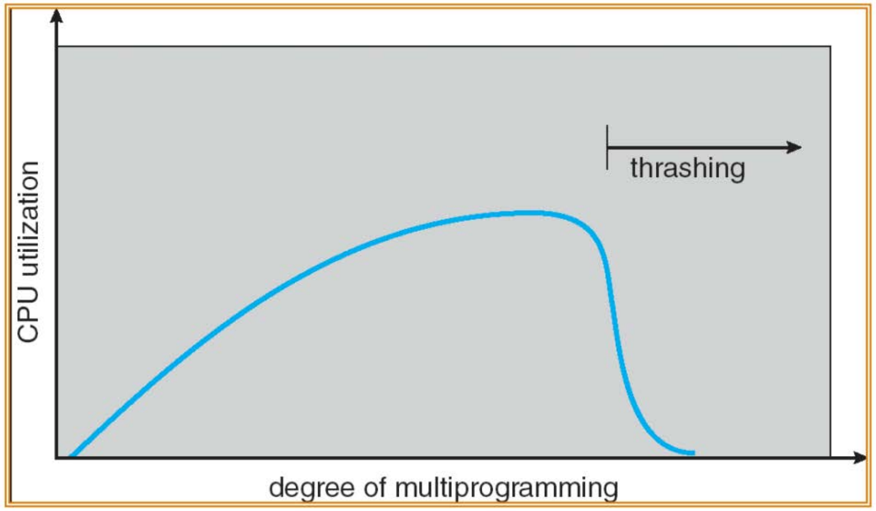
Global vs. Local Allocation

- Global replacement – process selects a replacement frame from the set of all frames; one process can take a frame from another
  - It is possible for processes to suffer page faults through no fault of theirs
  - However, improves system throughput
- Local replacement – each process selects from only its own set of allocated frames
  - May not use free space in the system

Thrashing

- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization
  - operating system thinks that it needs to increase the degree of multiprogramming because of low CPU utilization
  - another process added to the system

- Thrashing ≡ a process is busy swapping pages in and out



- https://blogs.technet.microsoft.com/markrussinovich/2008/11/17/pushing-the-limits-of-windows-virtual-memory/
- https://practice.geeksforgeeks.org/problems/what-is-compaction-is-os
- https://msdn.microsoft.com/en-us/library/windows/desktop/aa366914(v=vs.85).aspx
- http://cc.ee.ntu.edu.tw/~farn/courses/OS/OS2010/slides/ch09.pdf
- https://web.cs.hacettepe.edu.tr/~abc/teaching/bbs646/slides/ch10.pdf