Memory Management

Memory Management

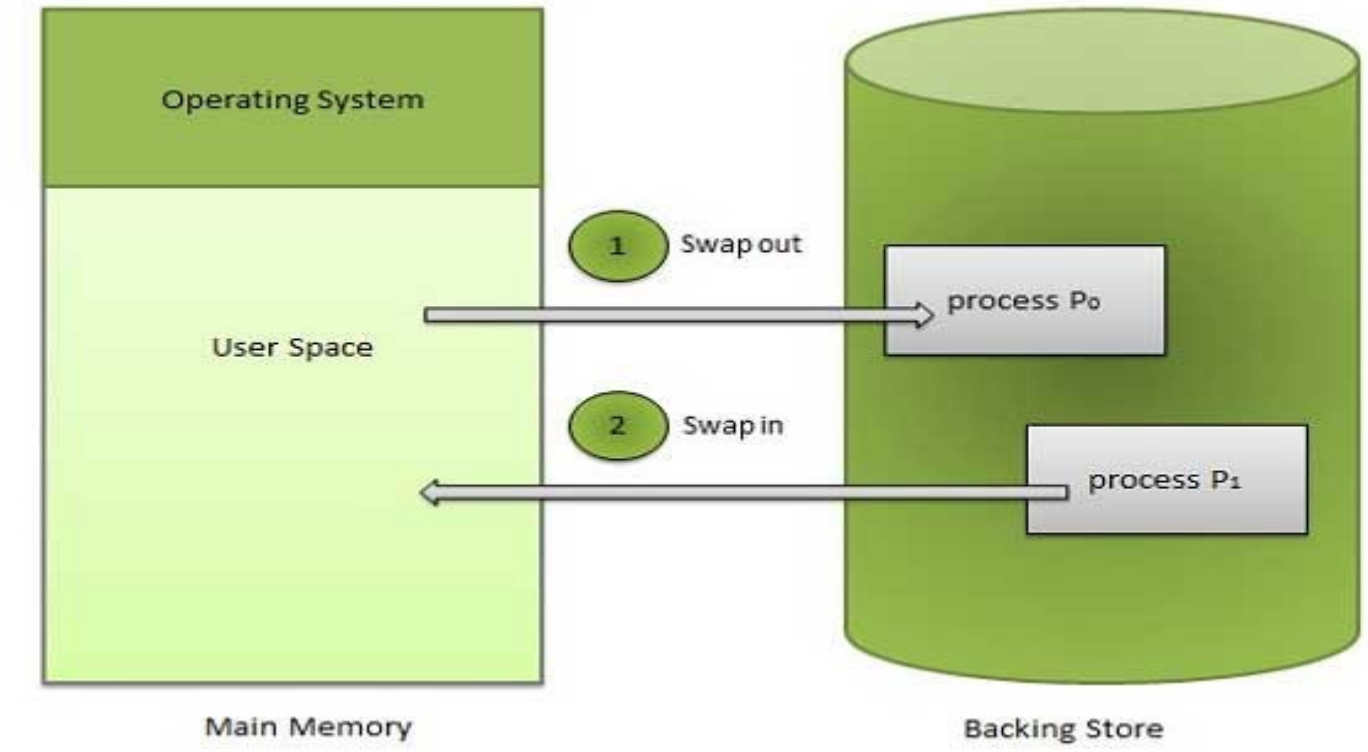
- Is the task carried out by the OS and hardware to accommodate multiple processes in main memory
- If only a few processes can be kept in main memory, then much of the time all processes will be waiting for I/O and the CPU will be idle
- Hence, memory needs to be allocated efficiently in order to pack as many processes into memory as possible

Level of Storage

- There are various levels of computer memory
  - Registers
  - Cache
  - Main Memory
  - Secondary Memory
  - Tertially Memory

Memory

- Main memory stores data and processes being executed by the cpu
- It consists of tiny cells which holds the data and processes
- Each cell has an address which identifies its location.
- The OS determines how the memory is utilised


Operating System
User Space
1
Swap out
process P0
2
Swap in
process P1
Main Memory
Backing Store

Process Execution Cycle

- Fetch – from memory or register
- Decode – interpret
- Execute – run
- Store – return back to memory/register

Logical Vs Physical Address

- **Logical address – generated by the CPU; also referred to as *virtual*** *address – identifies a process in CPU*
- **Physical address – Actual address of a cell unit**

OS maps logical address in physical address

MEMORY MANAGEMENT FUNCTIONS

1. Keep track of what parts of memory are in use. Ensure efficient memory utilization

2. Allocate memory to processes when needed.

3. Deallocate when processes are done.

4. Swapping, or paging, between main memory and disk, when disk is too small to hold all current processes

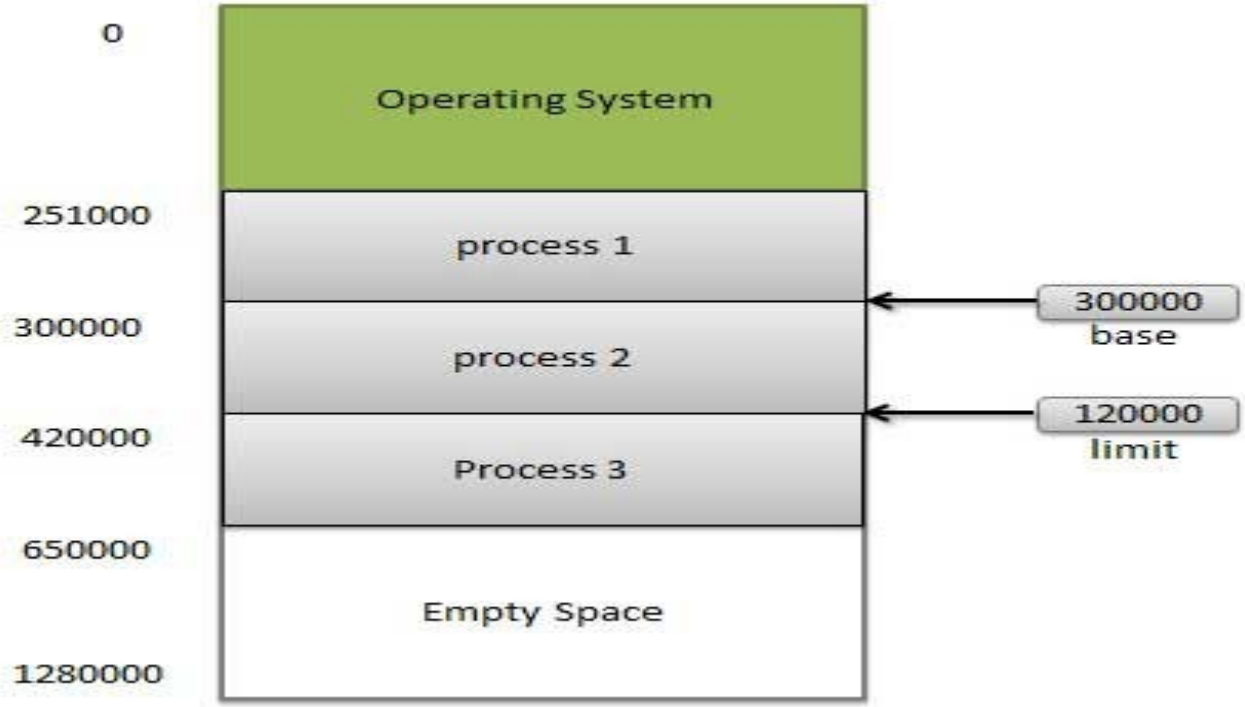
5. Relocation

- programmer cannot know where the program will be placed in memory when it is executed
- a process may be (often) relocated in main memory due to swapping
- memory references in code (for both instructions and data) must be translated to actual physical memory address

6. Protection

- processes should not be able to reference memory locations in another process without permission
- impossible to check addresses at compile time in programs since the program could be relocated
- address references must be checked at run time by hardware

- Memory management provides protection by using two registers, a base register and a limit register. The base register holds the smallest legal physical memory address and the limit register specifies the size of the range. For example, if the base register holds 300000 and the limit register is 1209000, then the program can legally access all addresses from 300000 through


0
Operating System
251000
process 1
300000
process 2
420000
Process 3
650000
Empty Space
1280000
300000
base
120000
limit

7. Sharing

- must allow several processes to access a common portion of main memory without compromising protection
  - cooperating processes may need to share access to the same data structure
  - better to allow each process to access the same copy of the program rather than have their own separate copy

8. Logical Organization

- users write programs in modules with different characteristics
  - instruction modules are execute-only
  - data modules are either read-only or read/write
  - some modules are private others are public
- To effectively deal with user programs, the OS and hardware should support a basic form of module to provide the required protection and sharing

9. Physical Organization

- secondary memory is the long term store for programs and data while main memory holds program and data currently in use
- moving information between these two levels of memory is a major concern of memory management (OS)
  - it is highly inefficient to leave this responsibility to the application programmer

Fragmentation

- As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes can not be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

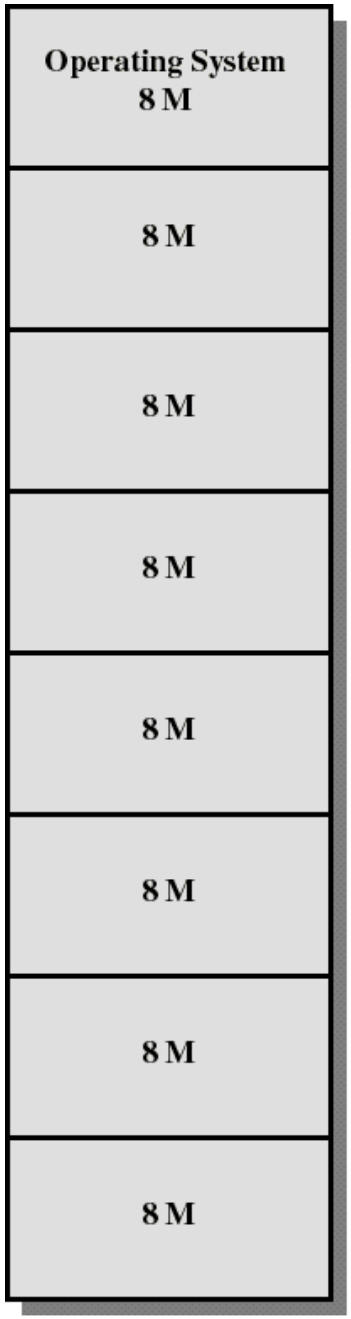
- **External fragmentation**
- Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it can not be used.
- **Internal fragmentation**
- Memory block assigned to process is bigger. Some portion of memory is left unused as it can not be used by another process.

Evolution Memory Management Schemes

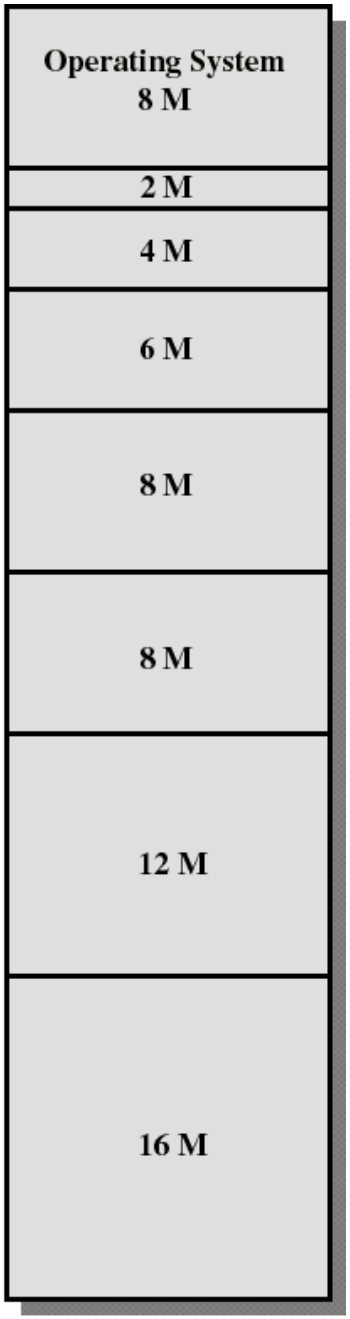
- Memory management schemes aims to accommodate as many processes as possible
- An executing process must be loaded in main memory
- The following memory management techniques have been used in OS over the years.
  - No management
  - fixed partitioning
  - dynamic partitioning
  - Virtual Memory
    - simple paging
    - simple segmentation

Fixed Partitioning

- Partition main memory into a set of non overlapping regions called partitions
- Partitions can be of equal or unequal sizes



Equal-size partitions



Unequal-size partitions

Fixed Partitioning

- any process whose size is less than or equal to a partition size can be loaded into the partition
- if all partitions are occupied, the operating system can swap a process out of a partition

Algorithms for allocating memory .

1. FIRST FIT - allocates the first hole found that is large enough - fast (as little searching as possible).

2. NEXT FIT - almost the same as First Fit except that it keeps track of where it last allocated space and starts from there instead of from the beginning - slightly better performance.

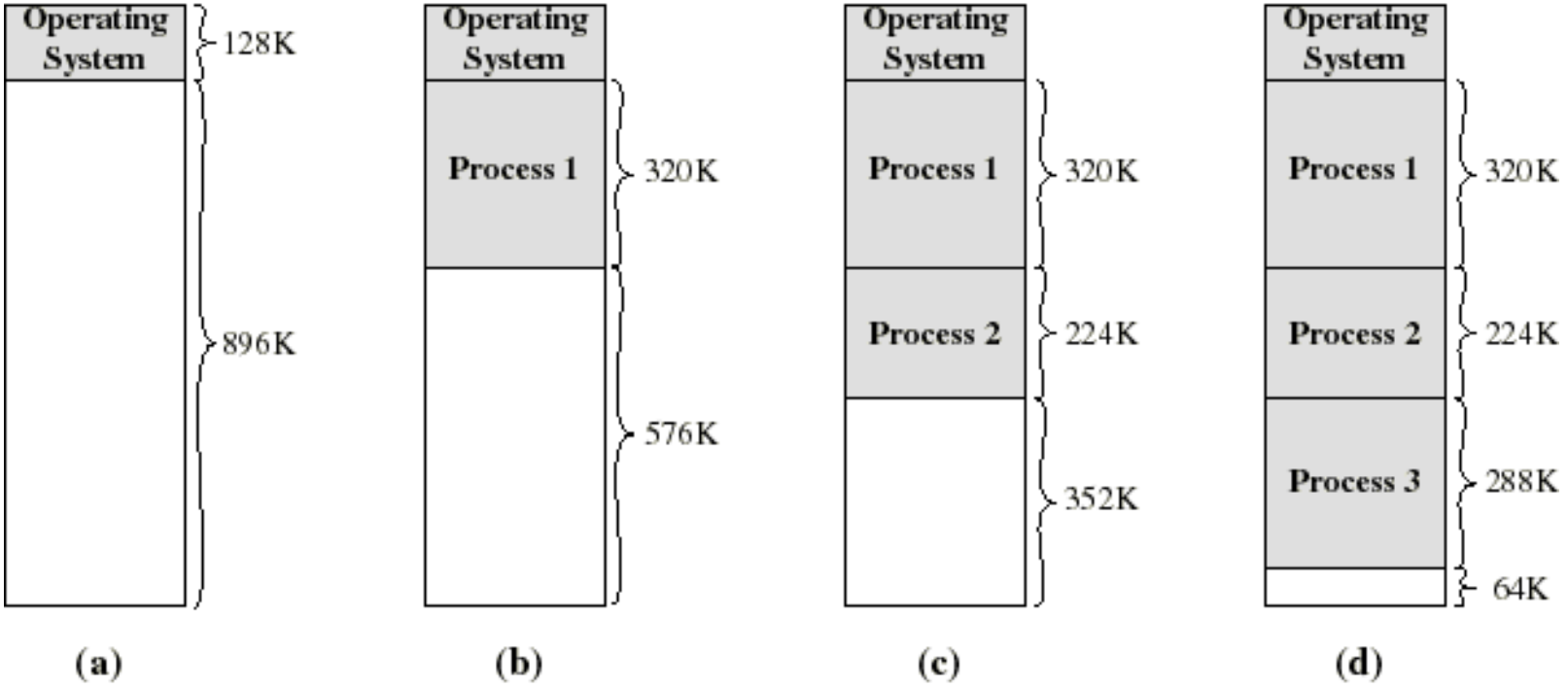
3. BEST FIT - searches the entire list looking for a hole that is closest to the size needed by the process - slow - also does not improve resource utilization because it tends to leave many very small (and therefore useless) holes.

4. WORST FIT - the opposite of Best Fit - chooses the largest available hole and breaks off a hole that islarge enouggh to be useful (I.e. hold another process) - in practice has not been shown to work better than others.

Dynamic Partitioning

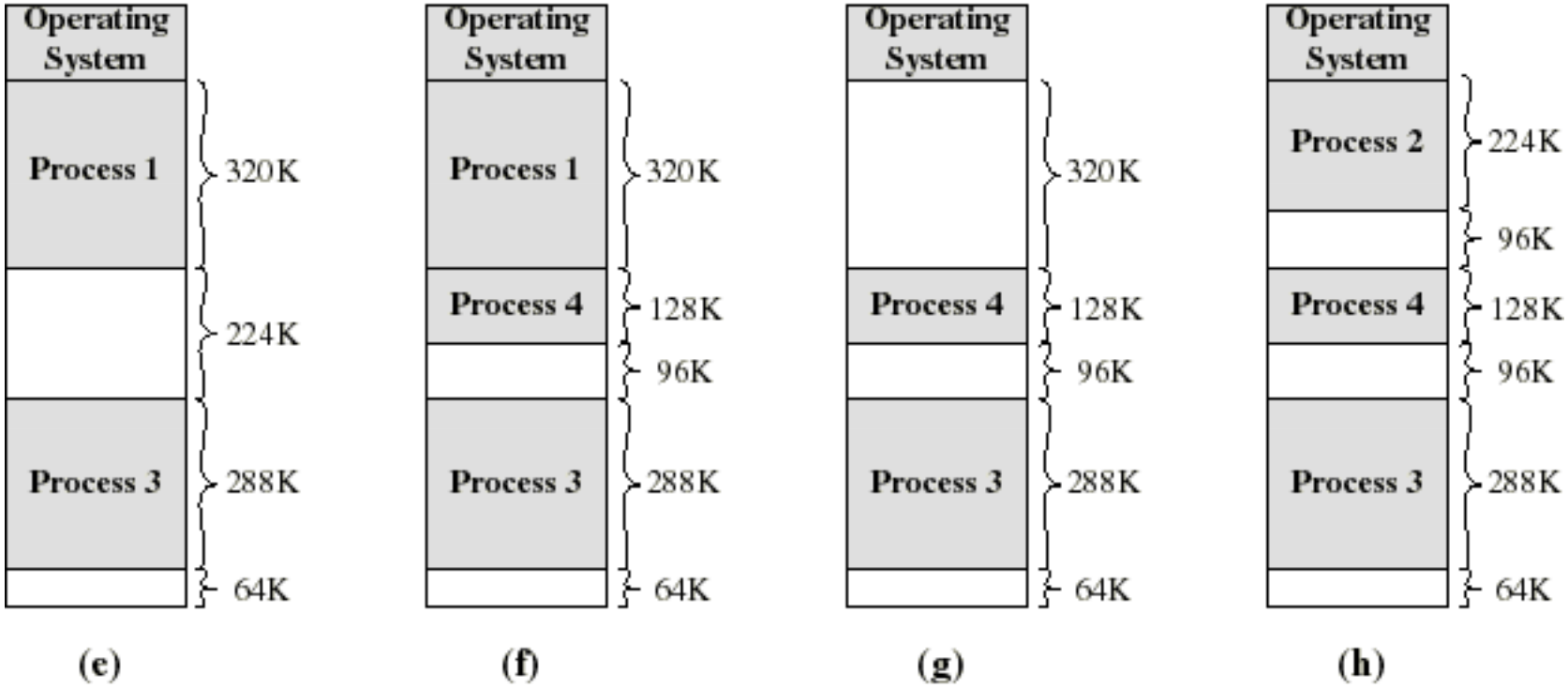
- Partitions are of variable length and number
- Each process is allocated exactly as much memory as it requires
- Used in IBM’s OS/MVT (Multiprogramming with a Variable number of Tasks)

Dynamic Partitioning: an example



- A hole of 64K is left after loading 3 processes: not enough room for another process
- Eventually each process is blocked. The OS swaps out process 2 to bring in process 4

Dynamic Partitioning: an example



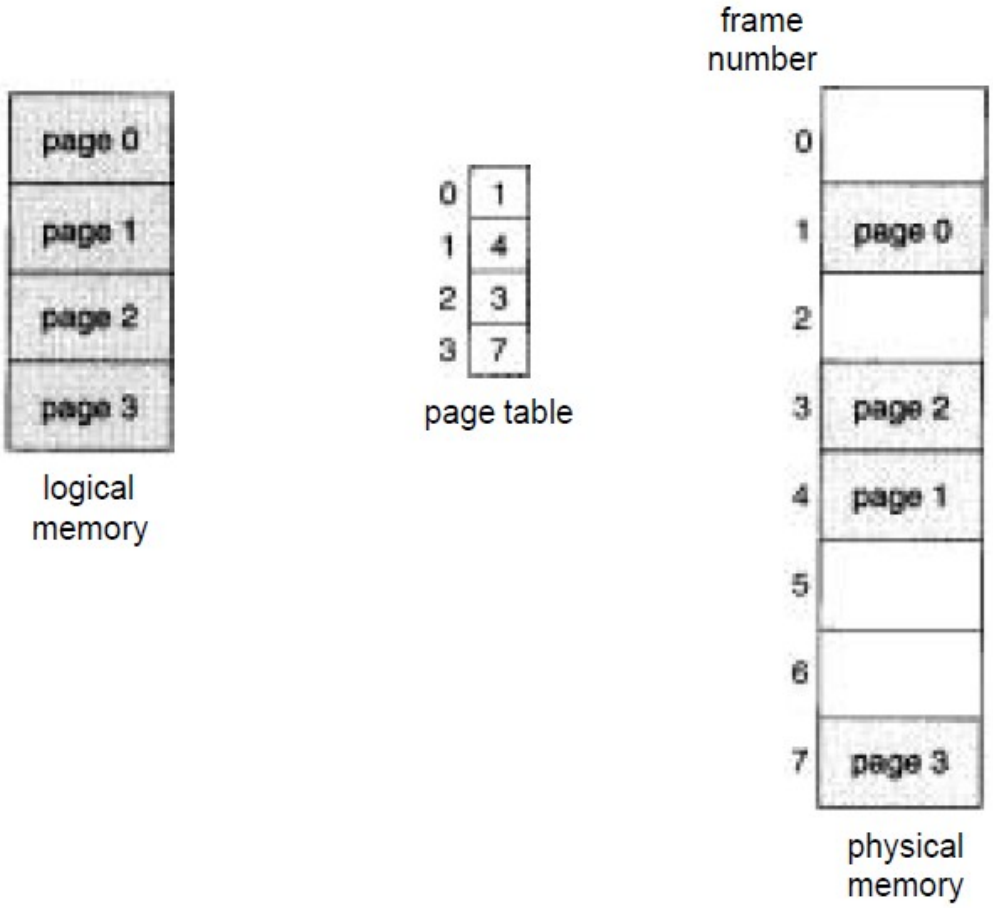
- another hole of 96K is created
- Eventually each process is blocked. The OS swaps out process 1 to bring in again process 2 and another hole of 96K is created...
- Compaction would produce a single hole of 256K

Virtual Memory

- The above schemes only make use of available memory.
- Virtual memory aims to make memory bigger than it really is by making use of secondary storage as an extension of main memory
- Implementation - paging, segmentation

Simple Paging

- Main memory is partition into equal fixed-sized chunks (of relatively small size) known as page frames
- Each process is also divided into chunks of the same size called pages
- The process pages can thus be assigned to the available frames (or page frames)
- Consequence: a process does not need to occupy a contiguous portion of memory


page 0
page 1
page 2
page 3
logical memory
0 1
1 4
2 3
3 7
page table
frame number
0
1 page 0
2
3 page 2
4 page 1
5
6
7 page 3
physical memory

- In a paging system, each process contains active pages in memory(physical memory) and inactive pages in logical memory.
- Inactive pages are swapped into memory when required. Similarly, active pages are swapped out (replaced) once executed.

Page Replacement Algorithms

- **Determine the order in which pages are swapped out from memory.**
  Replacement takes the following approach. If no frame is free, we find one
  that is not currently being used and free it.
- **First In First Out Algorithm**: when a new page must be brought in, replace the page that has been in memory the longest. Seldom used: even though a page has been in memory a long time, it may still be needed frequently.
- **Second Chance Algorithm**: this is a modification of FIFO. The Referenced bit of the page that has been in memory longest is checked before that page is automatically replaced. If the R bit has been set to 1, that page must have been referenced during the previous clock cycle. That page is placed at the rear of the list and its R bit is reset to zero. A variation of this algorithm, the '**clock**' algorithm keeps a pointer to the oldest page using a circular list. This saves the time used in the Second Chance Algorithm moving pages in the list

- **Not Recently Used Algorithm (NRU)** is a practical algorithm that makes use of the bits ‘Referenced’ and ‘Modified’. These bits are updated on every memory reference and must be set by the hardware. On every clock cycle the operating system can clear the R bit. This distinguishes those pages that have been referenced most recently from those that have not been referenced during this clock cycle. The combinations are:

(0) not referenced and not modified

(1) not referenced, modified

(2) referenced, not modified

(3) referenced, modified

NRU randomly chooses a page from the lowest class to remove

- **Least Recently Used Algorithm** (LRU) - keep track of each memory reference made to each page by some sort of counter or table. Choose a page that has been unused for a long time to be replaced. This requires a great deal of overhead and/or special hardware and is not used in practice. It is simulated by similar algorithms:
- **Not Frequently Used** - keeps a counter for each page and at each clock interrupt, if the R bit for that page is 1, the counter is incremented. The page with the smallest counter is chosen for replacement. What is the problem with this?

A page with a high counter may have been referenced a lot in one phase of the process, but is no longer used. This page will be overlooked, while another page with a lower counter but still being used is replaced.

Aging -a modification of NFU that simulates LRU very well. The counters are shifted right 1 bit before the R bit is added in. Also, the R bit is added to the leftmost rather than the rightmost bit. When a page fault occurs, the page with the lowest counter is still the page chosen to be removed. However, a page that has not been referenced for a while will not be chosen. It would have many leading zeros, making its counter value smaller than a page that was recently referenced.

Page Fault

- A page fault is said to occur in a page referenced is not available in memory.
- When a page fault occurs, the current process is halted and transfer into memory is initiated

How is a page fault actually handled?

1. Trap to the operating system (also called page fault interrupt).

2. Save the user registers and process state; i.e. process goes into waiting state.

3. Determine that the interrupt was a page fault.

4. Check that the page reference was legal and, if so, determine the location of the page on the disk.

5. Issue a read from the disk to a free frame and wait in a queue for this device until the read request is serviced. After the device seek completes, the disk controller begins the transfer of the page to the frame.

6. While waiting, allocate the CPU to some other user.

7. Interrupt from the disk occurs when the I/O is complete. Must determine that the interrupt was from the disk.

8. Correct the page table /other tables to show that the desired page is now in memory.

9. Take process out of waiting queue and put in ready queue to wait for the CPU again.

Simple Segmentation

- Each program is subdivided into blocks of non-equal size called segments
- When a process gets loaded into main memory, its different segments can be located anywhere
- Each segment is fully packed with instructs/data: no internal fragmentation
- There is external fragmentation; it is reduced when using small segments

Thrashing

- CPU utilization increases with increase in degree of multiprogramming.
- However, beyond a certain limit of multiprogramming CPU utilization suddenly starts to decrease.
- Reason - Thrashing

Simple Segmentation

- In contrast with paging, segmentation is visible to the programmer
  - provided as a convenience to organize logically programs (ex: data in one segment, code in another segment)
  - must be aware of segment size limit
- The OS maintains a segment table for each process. Each entry contains:
  - the starting physical addresses of that segment

Working Set Theory

The set of pages that a process is currently using is called its '**working set**'. If the entire working set is in memory, there will be no page faults.

If not, each read of a page from disk may take 10 milliseconds (.010 of a second). Compare this to the time it takes to execute an instruction: a few nanoseconds (.000000002 of a second). If a program has page faults every few instructions, it is said to be '**thrashing**'. Thrashing is happening

Page Replacement Algorithms (cont)

The **Working Set Algorithm** keeps track of a process' 'working set' and makes sure it is in memory before letting the process run. Since processes are frequently swapped to disk, to let other processes have CPU time, pure demand paging would cause so many page faults, the system would be too slow.

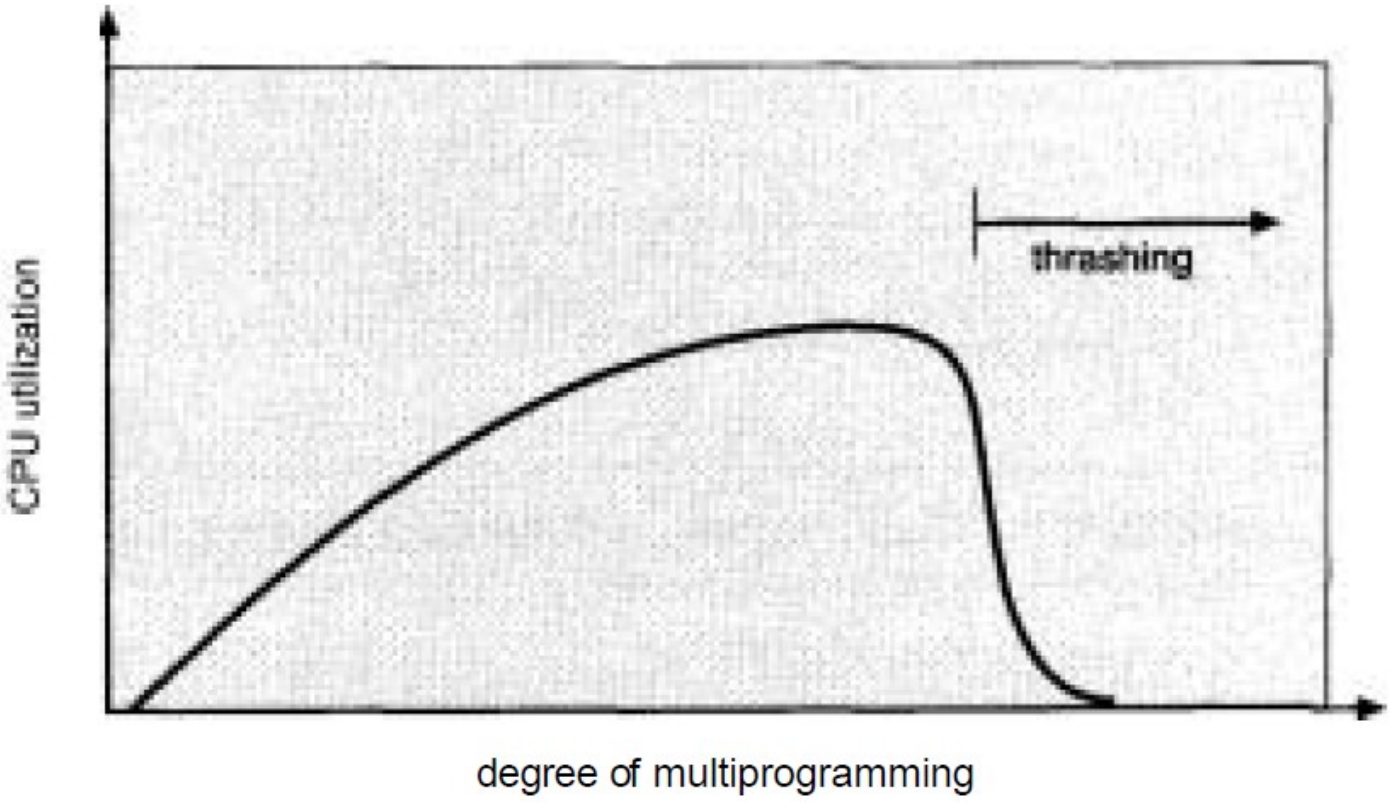
Ex. A program using a loop that occupies 2 pages and data from 4 pages, may reference all 6 pages every 1000 instructions. A reference to any other page may be a million instructions earlier

Comparison of paging and

Consideration	Paging	Segmentation
Need the programmer be aware that this technique is being used?	No	Yes
How many linear address spaces are there?	1	Many
Can the total address space exceed the size of physical memory?	Yes	Yes
Can procedures and data be distinguished and separately protected?	No	Yes
Can tables whose size fluctuates be accommodated easily?	No	Yes
Is sharing of procedures between users facilitated?	No	Yes
Why was this technique invented?	To get a large linear address space without having to buy more physical memory	To allow programs and data to be broken up into logically independent address spaces and to aid sharing and protection

Thrashing

- A process is thrashing if it is spending more time paging than executing.
- As the degree of multiprogramming increases, CPU utilization also increases, although more slowly,until a maximum is reached.
- If the degree of multiprogramming is increased even further, thrashing sets in and CPU utilization drops sharply


CPU utilization
thrashing
degree of multiprogramming

- The more the number of processes in memory, the less the number of pages for each process.
- The likelihood of a page fault occuring becomes very high and the missing page has to be swapped in.
- System spends more time paging than doing actual work
- This high paging activity is called **thrashing.**

Working Set Theory

- To avoid thrashing, a process is required to have a minimum number of pages in memory – Working Set.
- If the number of frames allocated to a low-priority process falls below the minimum number required by the computer architecture, we must suspend that process' execution.
- We should then page out its remaining pages, freeing all its allocated frames.

THE END