Virtual Memory

- It is a memory management scheme where only a portion of the virtual address space of a resident process may actually be loaded into physical memory.
- The sum of the virtual address spaces of active processes in a virtual memory system can exceed the capacity of the available physical memory provided that the physical memory is large enough to hold a minimum amount of the address space of each active process.

- Virtual memory can be accomplished by maintaining an image of the entire virtual address space of a process on secondary storage and by bringing its sections into main memory when needed.
- The choice of which sections to bring in, when to bring in and where to place them is made by OS. The allocation of physical memory is on demand basis.

- Demand paging
  - Do not require all pages of a process in memory
  - Bring in pages as required
- Page fault
  - Required page is not in memory
  - Operating System must swap in required page
  - May need to swap out a page to make space
  - Select page to throw out based on recent history

- We do not need all of a process in memory for it to run
- We can swap in pages as required
- So - we can now run processes that are bigger than total memory available!
- Main memory is called real memory
- User/programmer sees much bigger memory - virtual memory

Illusion of larger memory has many advantages.

1. Programmers are practically relieved of the burden of trying to fit a program into limited memory.

2. Some program may run without reprogramming or recompilation on systems with significantly different capacities of installed memory.

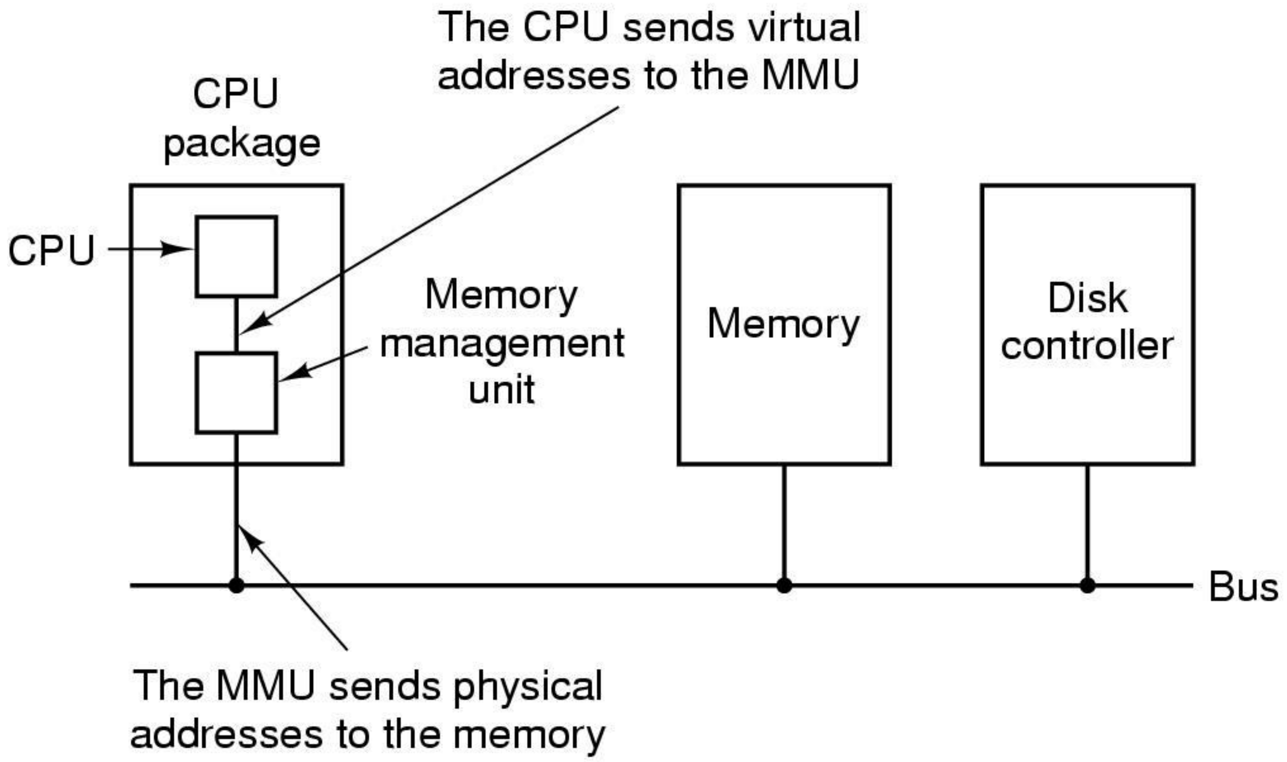
3. A process may be loaded into a space of arbitrary size. This may be used to reduce external fragmentation without the need to change the scheduled order of process execution.

4. OS can speed up the execution of important programs by allocating more memory.

5. Degree of multiprogramming can be increased by reducing the real memory for the resident processes.

The maximum program execution speed of a virtual memory system can be equal but never exceed the execution speed of the same program with virtual memory turned off.

Virtual Memory


The CPU sends virtual addresses to the MMU
CPU package
CPU
Memory management unit
Memory
Disk controller
Bus
The MMU sends physical addresses to the memory

The position and function of the MMU

Virtual memory can be implemented as an execution of paged or segmented memory management or as a combination of both.

Accordingly address translation is performed by means of PT, SDT or both.

Process of Address Mapping:

Assume Virtual memory is implemented using paging memory management.

Let Virtual address space be V = | 0,1, ...V-1| and Physical address space be M = |0,1,.. M-1|

At any time the mapping hardware must realize the function f : V-> M such that

f(x) = r if item x is in physical memory at location r else missing item exception if item x is not in physical memory.

- The detection of missing item is done by checking the presence indicator bit of each entry of PTs. Before loading the process OS clears all presence bits in the related PT.
- When pages are moved to main memory on demand, corresponding presence bits are set. When a page is evicted from main memory its presence bit is reset.
- The address translation hardware checks the presence bit during the mapping of each memory reference. If the bit is set, the mapping is completed as in paging scheme.

- If the bit is not set then the hardware generates a missing item exception which is referred as page fault in paging systems.
- When a running process experiences a page fault then the process is suspended until the missing page is brought to the main memory.
- Since the disk access time is longer than memory access time, another process is scheduled by OS.
- The disk address of the faulted page is provided by the File Map table (FMT). FMT contains secondary storage addresses of all pages.
- One FMT is maintained for each active process and has the number of entries identical to PT of the process. OS uses the virtual page number to index the FMT and to obtain the related disk address.

- Virtual memory needs more hardware provisions compared to paging and segmentation.
- One instruction may require several pages.
- For example, a block move of data. In some cases page fault may occur during part way through an operation and may have to undo what was done.
- Example: an instruction crosses a page boundary.

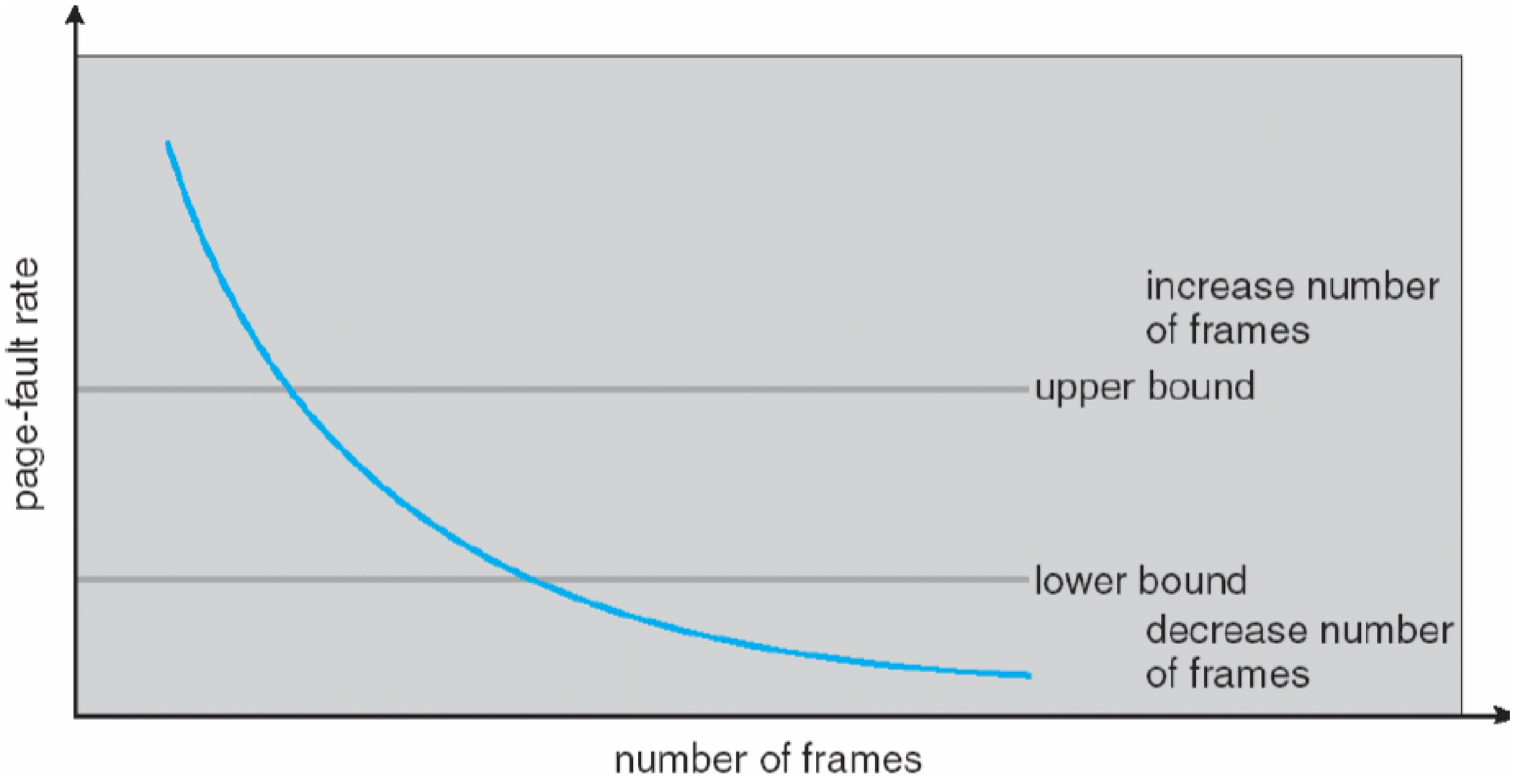
Management of Virtual Memory

- The allocation of subset of page frames to the virtual address space of a process requires the incorporation of certain policies into the virtual memory manager.
- Allocation policy: How much real memory to allocate to each process?
- Fetch Policy: What are the items to be brought and when to bring them?
- Placement policy: Where to place the incoming item?
- Replacement Policy: If there is no free memory, which item is to be evicted so that the new item can be stored in the real memory.

Allocation Policy

- OS should determine how many page frames should be allocated to a process.
- If more page frames are allocated then the advantages are reduced page fault frequency and improved turnaround time.
- If too few pages are allocated to a process, then page fault frequency and turnaround times may deteriorate to unacceptable levels and also in systems supporting restarting of faulted instructions page fault frequency may increase.
- The allocation module may fix two thresholds lower and upper for each process. The actual page fault rate experienced by a running process may be measured and recorded in PCB.
- When a process exceeds the upper threshold then more page frames will be allocated. If the process page fault rate reaches the lower threshold then allocation of new page frames may be stopped

Page fault rate vs allocation of page frames



- The fetch policy is mostly on demand and *Demand paging* is the most common virtual memory management system.
- It is based on the locality model of program execution. As a program executes, it moves from locality to locality.
- *Locality* is defined as a set of pages actively used together. A program is composed of several different localities which may overlap.
- For example, a small procedure when called, defines a new locality. A while-do loop when being executed defines a new locality.

- The Placement policy follows the underlying memory management scheme i.e. paging or segmentation.

Replacement Policy

- A page replacement algorithm *determines how the victim page (the page to be replaced) is selected when a page fault occurs*. find some page in memory, but not really in use, swap it out.
- There is a possibility that same page may be brought into memory several times. The aim is *to minimize the page fault rate*.

Steps in handling page faults including page replacement are:

1. Check the PT and FMT of the process, to determine whether the reference is a valid or an invalid memory access.

2. If the reference is invalid then the process is terminated; if it is valid and page have not yet brought in, find the location of the desired page on the disk.

3. Find a free frame. If there is a free frame use it. Otherwise, use a page-replacement algorithm to select a victim frame. Write the victim page to the disk; change the PT and MMT accordingly.

4. Schedule a disk operation to read the desired page into the newly allocated frame.

5. When the disk read is complete, modify the FMT, PT and MMT to indicate that the page is now in memory.

6 Restart the user process.

Page Replacement Algorithms

- Page fault forces choice
  - which page must be removed
  - make room for incoming page

- Modified page must first be saved
  - unmodified just overwritten

- Better not to choose an often used page
  - will probably need to be brought back in soon

- The efficiency of a page replacement algorithm is evaluated by running it on a particular string of memory references and computing the number of page faults.
- Reference strings are either generated randomly, or by tracing the paging behaviour of a system and recording the page number for each logical memory reference.
- The performance of a page replacement algorithm is evaluated by running it on a particular string of memory references and computing the number of page faults.

First-In-First-Out (FIFO)

- A FIFO algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. It is easy to implement.
- A FIFO queue/ list is used to hold all pages in memory. The page at the head of the queue is replaced. When a page is brought into memory, it is inserted at the tail of the queue.
- FIFO algorithm is easy to understand and to program.
- Drawback is Belady’s Anomaly - When the number of page frames allotted increases the page fault also increases for some cases.

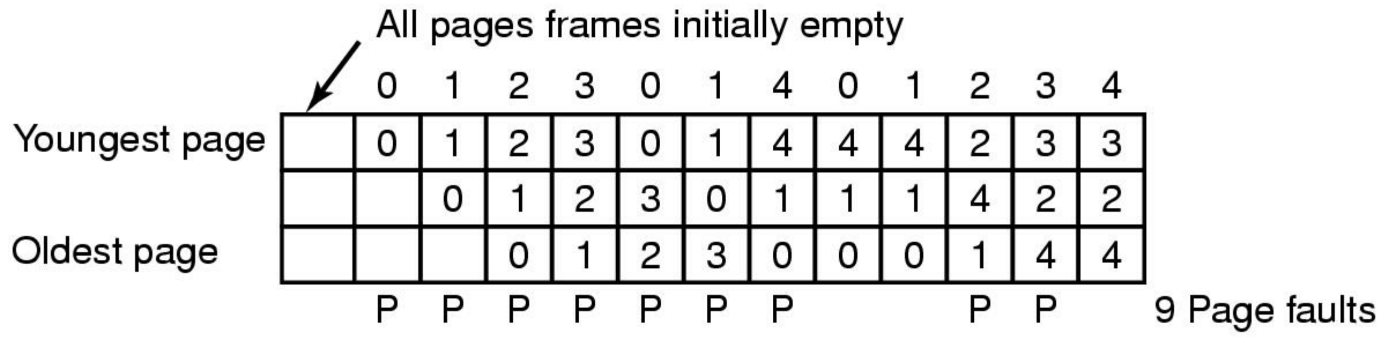
Example: FIFO

- Suppose we have 3 page frames, 4 virtual pages, and following reference stream: A B C A B D A D B C B

Ref: Page:	A	B	C	A	B	D	A	D	B	C	B
1	A					D				C	
2		B					A				
3			C						B		

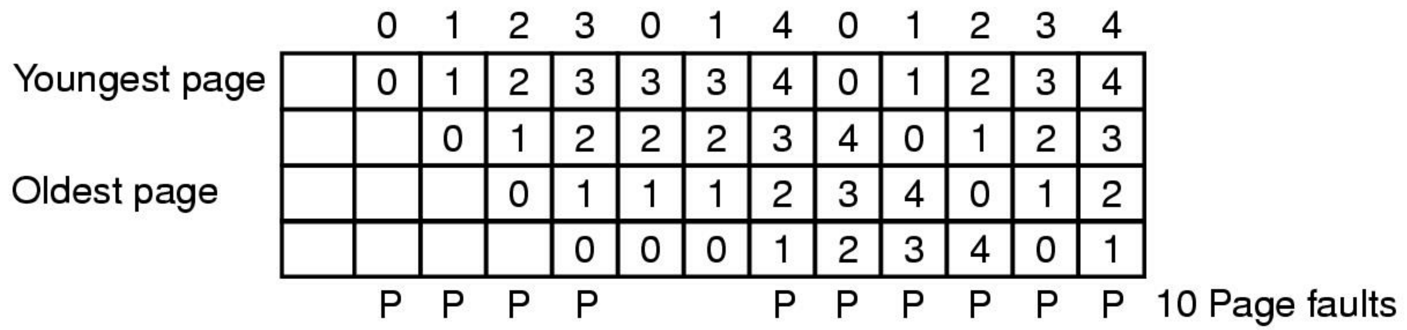
- FIFO: 7 faults.
- When referencing D, replacing A is bad choice, since need A again right away

Modeling Page Replacement Algorithms
Belady's Anomaly


All pages frames initially empty

		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	0	1	4	4	4	2	3	3	
			0	1	2	3	0	1	1	1	4	2	2	
Oldest page				0	1	2	3	0	0	0	1	4	4	
		P	P	P	P	P	P	P			P	P		9 Page faults

(a)



		0	1	2	3	0	1	4	0	1	2	3	4	
Youngest page		0	1	2	3	3	3	4	0	1	2	3	4	
			0	1	2	2	2	3	4	0	1	2	3	
Oldest page				0	1	1	1	2	3	4	0	1	2	
					0	0	0	1	2	3	4	0	1	
		P	P	P	P			P	P	P	P	P	P	10 Page faults

(b)

- FIFO with 3 page frames
- FIFO with 4 page frames
- *P*'s show which page references show page faults

Least Recently Used (LRU)

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement. Easy to implement, keep a list, replace pages by looking back into time.
- The OS using this method, has to associate with each page, the time it was last used which means some extra storage.
- In the simplest way, the OS sets the reference bit of a page to "1" when it is referenced. This bit will not give the order of use but it will simply tell whether the corresponding frame is referenced recently or not. The OS resets all reference bits periodically.

LRU Algorithm Implementations

Counter implementation:

- A time-of-use field is associated with each page-table entry, and a logical clock or counter is added to the CPU.
- The clock is incremented for every memory reference. Whenever a reference to a page is made, the content of the clock register is copied to the time-of-use field in the page table for that page.
- The page with the smallest time value is replaced. It requires a search of the page table to find LRU page and a write to memory for each memory access.

Stack implementation:

- A stack is used to store page numbers. Whenever a page is referenced, it is removed from the stack and put on the top.
- Top of the stack is always the mostly recently used page and the bottom is the LRU page.
- It is implemented by a double linked list with a head and tail pointer because entries must be removed from the middle of the stack. The tail pointer points to the bottom of the stack (which is LRU page).

Consider the page reference string A B C B B A D A B F B. If LRU page replacement is followed and the number of page frames allocated is 3, find the total number of page faults.

Total number of page faults = 5

A	B	C	B	B	A	D	A	B	F	B
A	A	A	A	A	A	A	A	A	A	A
	B	B	B	B	B	B	B	B	B	B
		C	C	C	C	D	D	D	F	F
*	*	*				*			*	

A	B	C	B	B	A	D	A	B	F	B
A	B	C	B	B	A	D	A	B	F	B
	A	B	C	C	B	A	D	A	B	F
		A	A	A	C	B	B	D	A	A

Optimal Page Replacement Algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. It has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. (Optimal but unrealizable)
- In this algorithm, the victim is the page which will not be used for the longest period. Estimate by logging page - use on previous runs of process although this is impractical
- It will never suffer from Belady’s anomaly.
- OPT is not possible to be implemented in practice because it requires future knowledge. However, it is used for performance comparison.

Page Buffering algorithm

- The replaced pages will first go to one of two page lists in main memory, depending whether it has been modified or not.
- The advantage of this scheme is that if in a short time, the page is referenced again, it may be obtained with little overhead.
- When the length of the lists surpasses a specific limit, some of the pages that went into the lists earlier will be removed and written back to secondary memory. Thus the two lists actually act as a cache of pages.
- Another advantage is the modified pages may be written out in cluster rather than one at a time, which significantly reduces the number of I/O operations and therefore the amount of time for disk access pool.

Counting Algorithms

It keeps a counter of the number of references that have been made to each page. Two schemes use this concept.

1. Least Frequently Used (LFU) Algorithm:

Page with the smallest count is the one which will be selected for replacement.

This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again

2. Most Frequently Used (MFU) Algorithm:

It is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

Second-Chance Algorithm/ Clock Algorithm

- The clock/ second chance policy is basically a variant of the FIFO policy, except that it also considers to some extent the last accessed times of pages.
- When a page has been selected, its reference bit is inspected. If the value is 0 replace the page. If the value is 1, then the page is given a second chance and move on to select the next FIFO page.

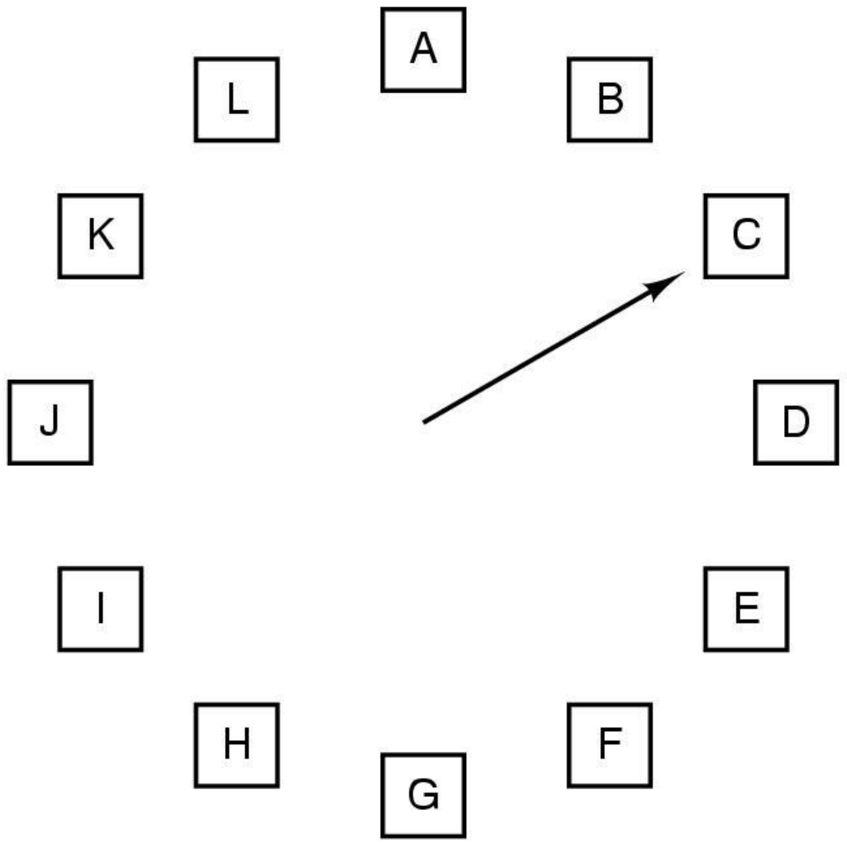
When a page gets a second chance:

1. Its reference bit is cleared.

2. Its arrival time is reset to the current time.

3. It will not be replaced until all other pages are replaced or given second chance.

The Clock Page Replacement Algorithm



When a page fault occurs, the page the hand is pointing to is inspected. The action taken depends on the R bit:

- R = 0: Evict the page
- R = 1: Clear R and advance hand

Enhanced Second-Chance Algorithm (Not Recently Used)

Each page has Reference bit, Modified bit : bits are set when page is referenced, modified. It uses the reference bit and the modify bit as an ordered pair.

With 2-bits, four classes are possible:

1. (0,0)– neither recently used nor modified (best page to replace).
2. (0,1)– not recently used but modified the page will need to be written out before replacement.
3. (1,0)– recently used but clean (probably will be used again).
4. (1,1)– recently used and modified probably will be used again and write out will be needed before replacing it.

The steps of the algorithm are as follows:

1. Scan the frame buffer and select the first frame with (0, 0) if any.

2. If step 1 fails, scan again, look for a frame with (0,1) and select it for replacement if such a frame exists. During the scan, the reference bits of 1 are set to 0.

3. If step 2 fails, all the reference bits in the buffer are 0. Repeat step 1 and if necessary step 2. This time, a frame will be definitely chosen to be replaced.

The recently accessed pages are given higher priority than those that have been modified, based on the consideration that though the latter need to be written back to secondary memory first before replacement, they still involve less overhead than probable reloading a recently accessed but replaced page.

Frame Allocation

In order to decide on the page replacement scheme of a particular reference string, the number of page frames available should be known.

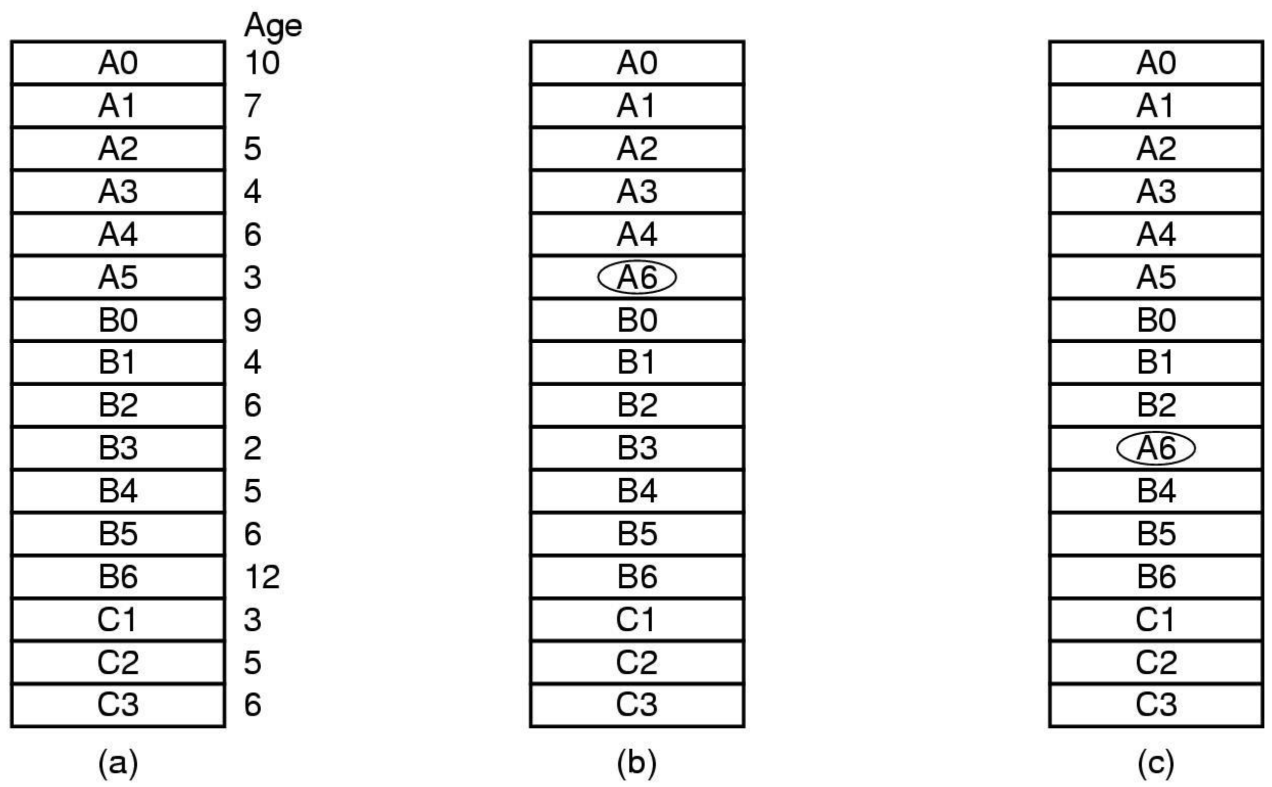
In page replacement, some frame allocation policies may be followed.

- Global Replacement: A process can replace any page in the memory.
- Local Replacement: Each process can replace only from its own reserved set of allocated page frames.

In case of local replacement, the operating system should determine how many frames should the OS allocate to each process.

The number of frames for each process may be adjusted by using two ways: Equal ; Proportional

Design Issues for Paging Systems
Local versus Global Allocation Policies (1)



- Original configuration
- Local page replacement
- Global page replacement

Thrashing

- Too many processes in too little memory
- Operating System spends all its time swapping
- Little or no real work is done
- Disk light is on all the time

- Solutions
  - Good page replacement algorithms
  - Reduce number of processes running
  - Fit more memory

Working Set Model

To prevent thrashing, a process must be provided with as many frames as it needs. For this, a model called *the working set model* is developed which depends on the locality model of program execution.

The pages used by a process within a window of time are called its working set. A parameter, Δ, called the working set window size is used. The set of pages in the last Δ page references is the working set of a process.

Choice of Δ is crucial. If Δ is too small, it will not cover the entire working set. If it is too large, several localities of a process may overlap.

The working set principle states that

1. A program should be run if and only if its working set is in memory.

2. A page may not be removed from memory if it is the member of the working set of a running process.

Protection and Sharing

- In virtual systems protection and sharing retains the characteristics of the underlying memory management systems such as paging or segmentation. However, the frequent moving of items between main memory and secondary memory may complicate the management of tables.
- If a portion of a shared object is selected for replacement then the concerned tables should be updated accordingly. All copies of the mapping information must be kept in synchrony and updated to reflect the changes of residence of shared objects.

Advantages:

User's point of view:

Illusion of large address space almost eliminates the considerations imposed by limited physical memory.

Automatic management of memory makes the program run faster.

OS's point of view:

Ability to vary the amount of physical memory in use by any program.

CPU utilization is increased and wastage of memory is reduced.

•

Disadvantages:

Complex hardware and software needed to support virtual memory.

Both time and space complexities are more when compared to other memory management schemes.

Higher table fragmentation.

Possibility of thrashing leads to complex page replacement algorithms.

Average Turnaround time is increased due to missing item exceptions.