Virtual Memory

Virtual Memory

- Allows Execution of Processes that may not be completely in Memory
- Separation of user logical memory from physical memory.
- Only part of the program needs to be in memory for execution.
- Logical address space can therefore be much larger than physical address space.
- Gives Illusion of infinite physical memory
- Need to allow pages to be swapped in and out.
- Virtual memory can be implemented via:
  - Demand paging
  - Demand segmentation

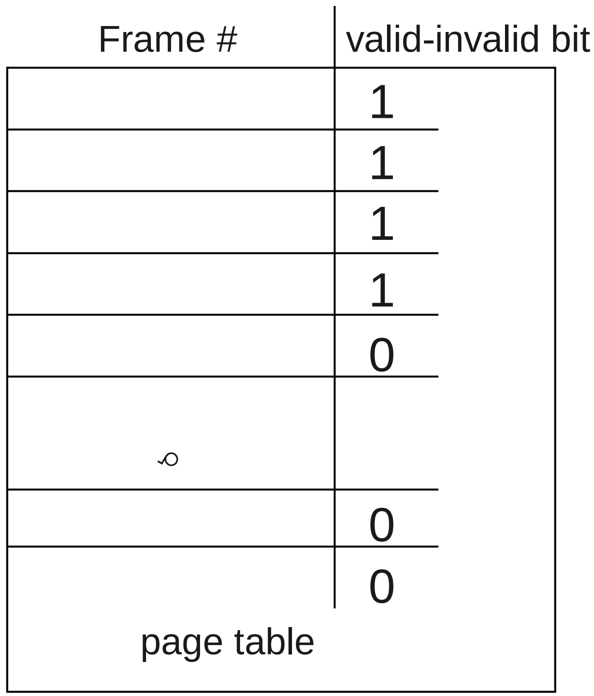
Demand Paging

- Bring a page into memory only when it is needed.
  - Less I/O needed
  - Less memory needed
  - Faster response
  - More users

- Page is needed ⇒ reference to it
  - invalid reference ⇒ abort
  - not-in-memory ⇒ bring to memory

Valid-Invalid Bit

With each page table entry a valid–invalid bit is associated
- (1 ⇒ in-memory, 0 ⇒ not-in-memory – Page Fault)
- Initially valid–invalid but is set to 0 on all entries.
- Example of a page table snapshot.



Page Fault

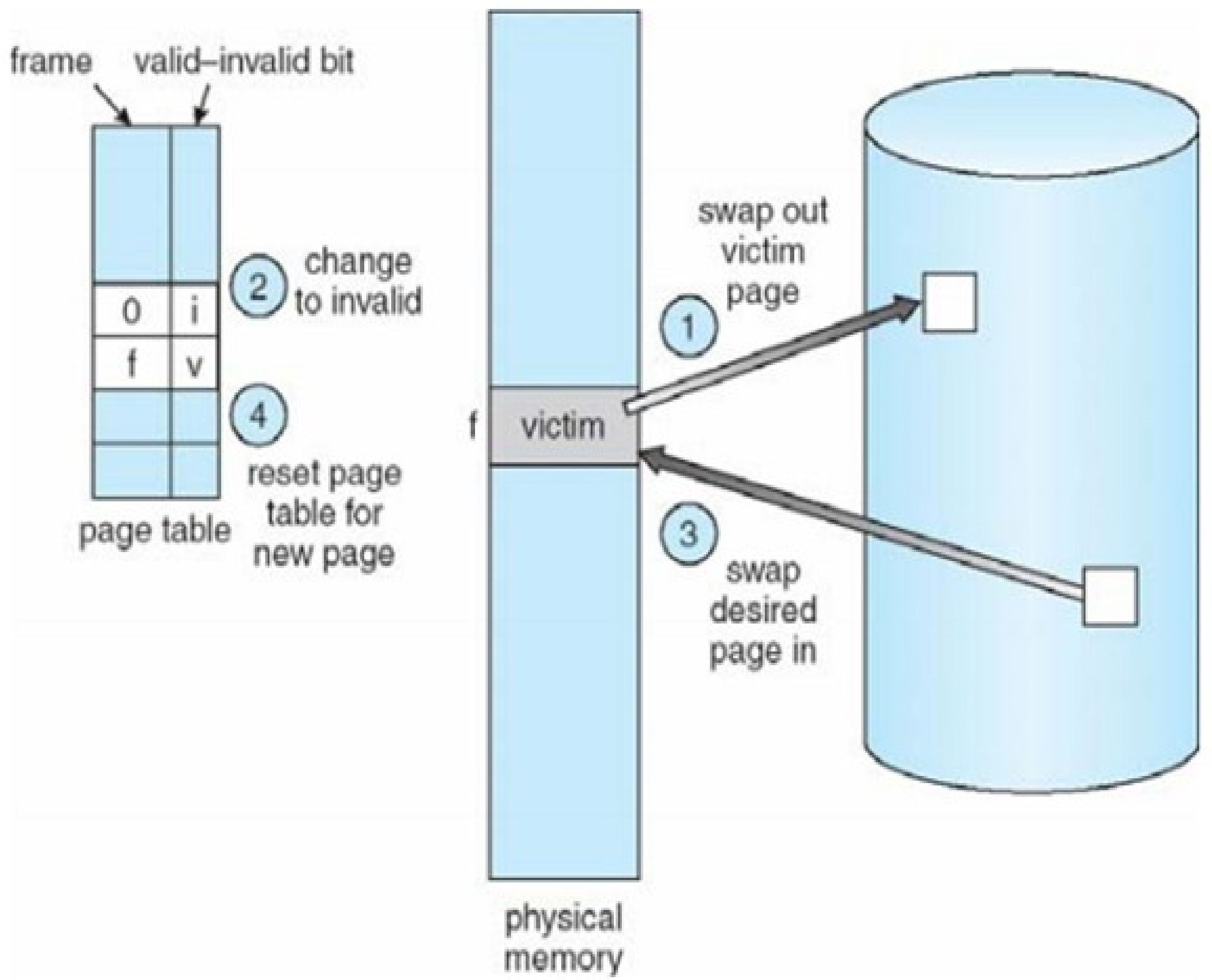
- If there is ever a reference to a page, first reference will trap to OS ⇒ page fault
- OS looks at another table to decide:
  - Invalid reference ⇒ abort.
  - Just not in memory.
- Get empty frame.
- Swap page into frame.
- Reset tables, validation bit = 1. Restart instruction

Page Replacement

- Page replacement – find some page in memory, but not really in use, swap it out.
  - algorithm
  - performance – want an algorithm which will result in minimum number of page faults.

- Same page may be brought into memory several times.

Page Replacement



Page Replacement Algorithms

- Want lowest page-fault rate.

- Evaluate algorithm by running it on a particular string of memory references (reference string) and computing the number of page faults on that string.

- In all our examples, the reference string is
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.

- Algorithms
  - FIFO (First In First Out)
  - LRU (Least Recently Used)
  - Optimal

First In First Out (FIFO) Algorithm

Reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	4	4	4	5	5	5	5	5	5
F2		2	2	2	1	1	1	1	1	3	3	3
F3			3	3	3	2	2	2	2	2	4	4
PF	1	2	3	4	5	6	7	7	7	8	9	9

Total page fault = 9

First In First Out (FIFO) Algorithm

Using 4 Frames

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	1	1	1	5	5	5	5	4	4
F2		2	2	2	2	2	2	1	1	1	1	5
F3			3	3	3	3	3	3	2	2	2	2
F4				4	4	4	4	4	4	3	3	3
PF	**1**	**2**	**3**	**4**	**4**	**4**	**5**	**6**	**7**	**8**	**9**	**10**

Total page fault = 10

FIFO Replacement – Belady's Anomaly
more frames $\Rightarrow$ less page faults

Advantages

- Very simple algorithm
- Easy to apply

Disadvantages

- Its facing Belady's anomaly
- Poor efficiency
- Not able to control page faults

Least Recently Used (LRU) Algorithms

Reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	4	4	4	5	5	5	3	3	3
F2		2	2	2	1	1	1	1	1	1	4	4
F3			3	3	3	2	2	2	2	2	2	5
PF	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**X**	**X**	**8**	**9**	**10**

Total page fault = 10

Using 4 Frames

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	1	1	1	1	1	1	1	1	5
F2		2	2	2	2	2	2	2	2	2	2	2
F3			3	3	3	3	5	5	5	5	4	4
				4	4	4	4	4	4	3	3	3
PF	**1**	**2**	**3**	**4**	**4**	**4**	**5**	**5**	**5**	**6**	**7**	**8**

Total page fault = 8

Least Recently Used (LRU) Algorithms

Advantages

- Very simple algorithm
- Easy to apply
- No. of Page Faults Decreased

Disadvantages

- Its facing Belady's anomaly for some reference strings
- Most of the times act as FIFO
- Not able to control page faults

Optimal Algorithm

Reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	1	1	1	1	1	1	3	3	1
F2		2	2	2	2	2	2	2	2	2	4	2
F3			3	4	4	4	5	5	5	5	5	5
PF	**1**	**2**	**3**	**4**	**4**	**4**	**5**	**5**	**5**	**6**	**7**	**8**

Total page fault = 7

Optimal Algorithm

Reference string:

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

F1	1	1	1	1	1	1	1	1	1	1	4	4
F2		2	2	2	2	2	2	2	2	2	2	2
F3			3	3	3	3	3	3	3	3	3	3
F4				4	4	4	5	5	5	5	5	5
PF	**1**	**2**	**3**	**4**	**4**	**4**	**5**	**5**	**5**	**5**	**6**	**6**

Total page fault = 6

Advantages

- Number of page faults are decreases
- Belady's anomaly exception is not occurs

Disadvantages

- Not gives proper result if there is not future reference string

Comparison

- The FIFO algorithm uses time when page was brought into memory

- The Optimal algorithm uses time when a page is to be used

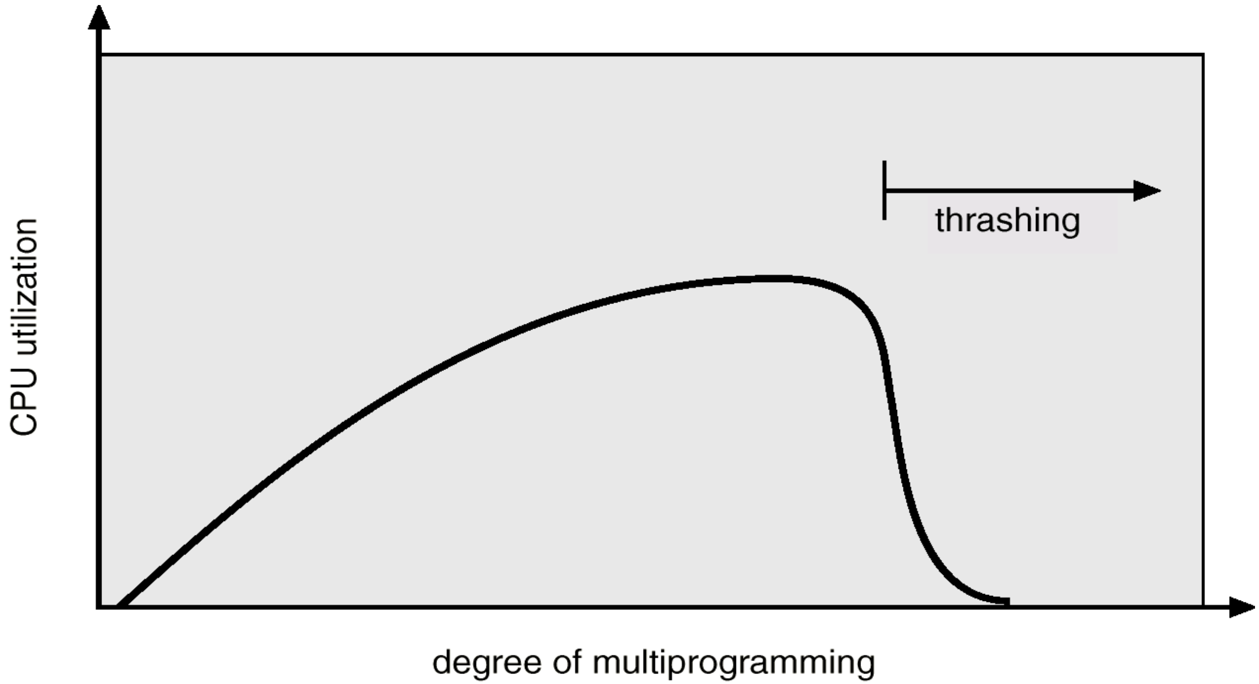
- LRU algorithm associates with each page the time of that page's last use.

Out of these three algorithms Optimal is best algorithm which gives less page fault

Thrashing

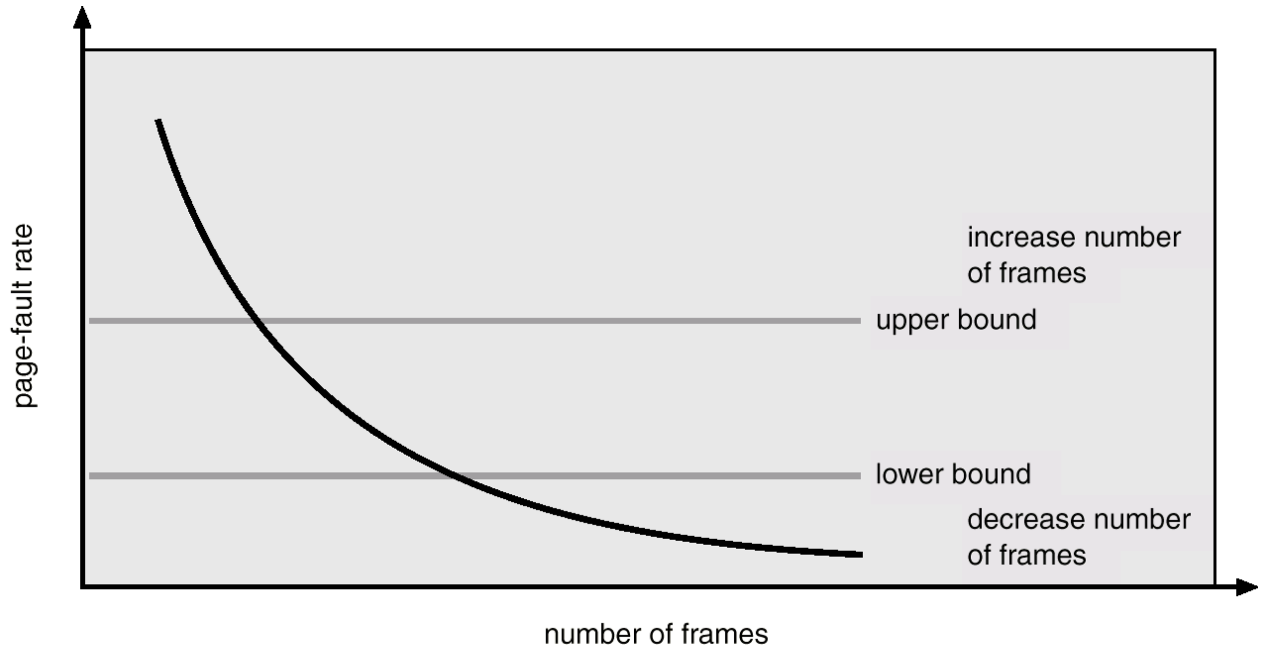
- If a process does not have “enough” pages, the page-fault rate is very high. This leads to:
  - low CPU utilization.
  - operating system thinks that it needs to increase the degree of multiprogramming.
  - another process added to the system.
- Thrashing $\equiv$ a process is busy swapping pages in and out.

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

Page Fault Frequency Scheme



- Establish “acceptable” page-fault rate.
  - If actual rate too low, process loses frame.
  - If actual rate too high, process gains frame.

Other Considerations

- Preparing
- Page size selection
  - fragmentation
  - table size
  - I/O overhead
  - locality