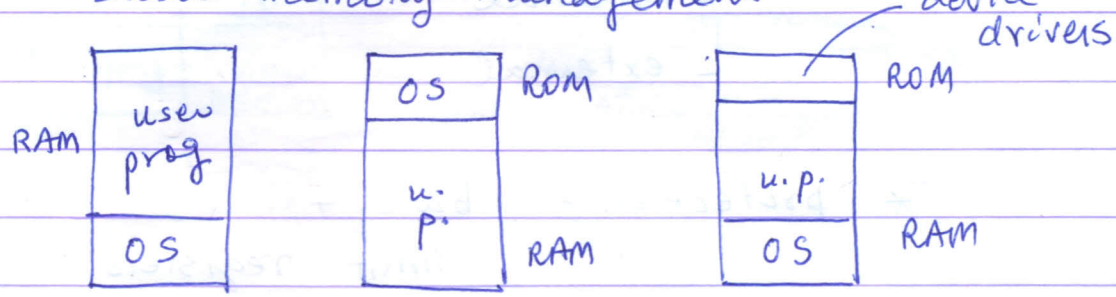
OS
VII

MEMORY MANAGEMENT

4.1 Basic Memory Management



* ⟶ (monoprogramming)

no swapping
no paging

* Multiprogramming with fixed partitions [OS/MFT]

↑ increases CPU utilization

but: logical address ≠ physical address

queue for each partition
(or)
one queue

* fragmentation
  - internal
  - external

* protection - base + limit registers

(or)

protection bits per page

*

* CPU utilization $= 1 - (1-p)^n$

n = degree of multiprogramming

sec. 4.1.4

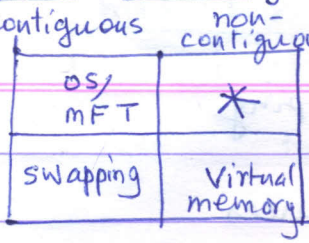
* contiguous vs non-contiguous allocation

logical page

(physical) page frame

unit of allocation : page

options for sharing main memory



do not use disk as backup*

use disk as backup*

for running program

* implementation using bit maps / linked lists

4.2 Swapping

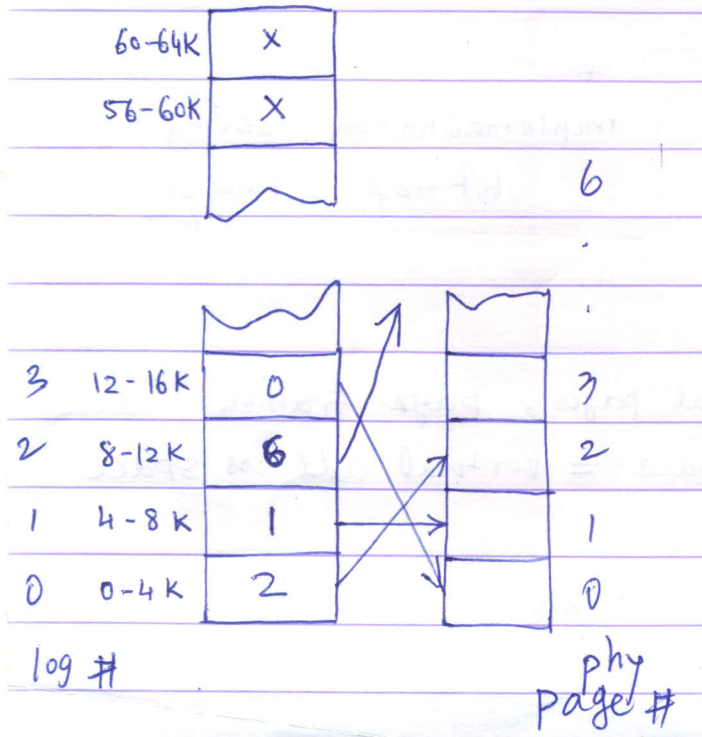
fig. 4-5

VIRTUAL MEMORY

- paging - logical pages, page frames
- logical address space = virtual address space

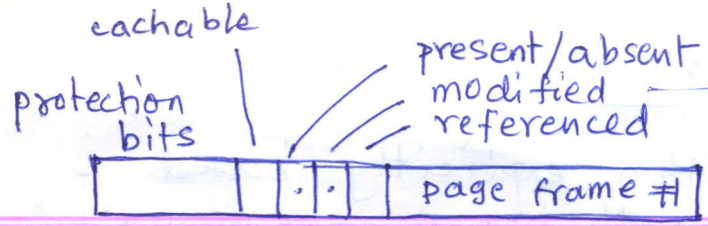
  fig. 4-10
- structure of page table

- issues to be addressed
  - size of page table
  - need for fast mapping
- locality of program references
- principle of caching



relation between virtual addresses and physical mem. addresses

[relate with fig. 4-11]



typical 32 bits

Translation lookaside buffers (TLBs)

- principle of cache memory applied to page tables
- uses associative memory for faster access

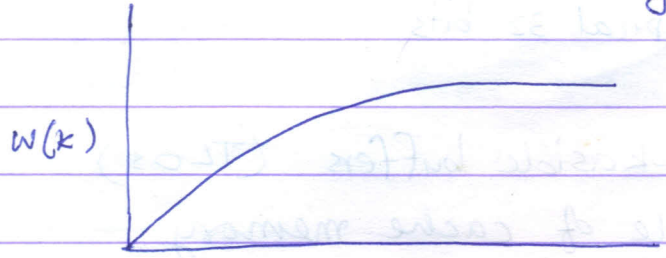
(fig 4-14)

4.4 PAGE REPLACEMENT ALGORITHMS

- optimal p.r.a.
- NRU
- LRU
- FIFO
- FIFO with second chance
- clock based approximation NFU
- aging

- Working set

algorithm explicitly based on the working set model



- prepaging
- thrashing

→ SHARING OF PAGES

SEGMENTATION → a bit later*

4.7.1 Operating system involvement with paging – self study

4.7.2 Page fault handling (see below)

On a page fault -

	1.	h/w traps to kernel
p handler	2.	process registers / state information saved.
		OS procedure called.
OS	3.	determine virtual page needed (logical)
"	4.	check for valid access.
"		check for free page frame. if not available,
"		run page replacement algorithm.
"	5.	if dirty, write back page to be replaced
"	6.	locate needed page on disk. initiate I/O
"		to bring (read) it in from disk.
"	7.	on completion of I/O, update page table
"	8.	back up faulting instruction, reset PC
"	9.	schedule faulting process. return to trap
"		handler
handler	10.	restore registers / state, return to user
		space to resume execution of process

SEGMENTATION

- Logical address space available to a program is segmented, with each segment running from address 0 to its upper limit
- programmer / system designer uses the different segments for different functions or groups of functions

- Intel 8086 processor had 4 segments
- Intel Pentium provides segmentation and paging

Fig 4-37 – self study