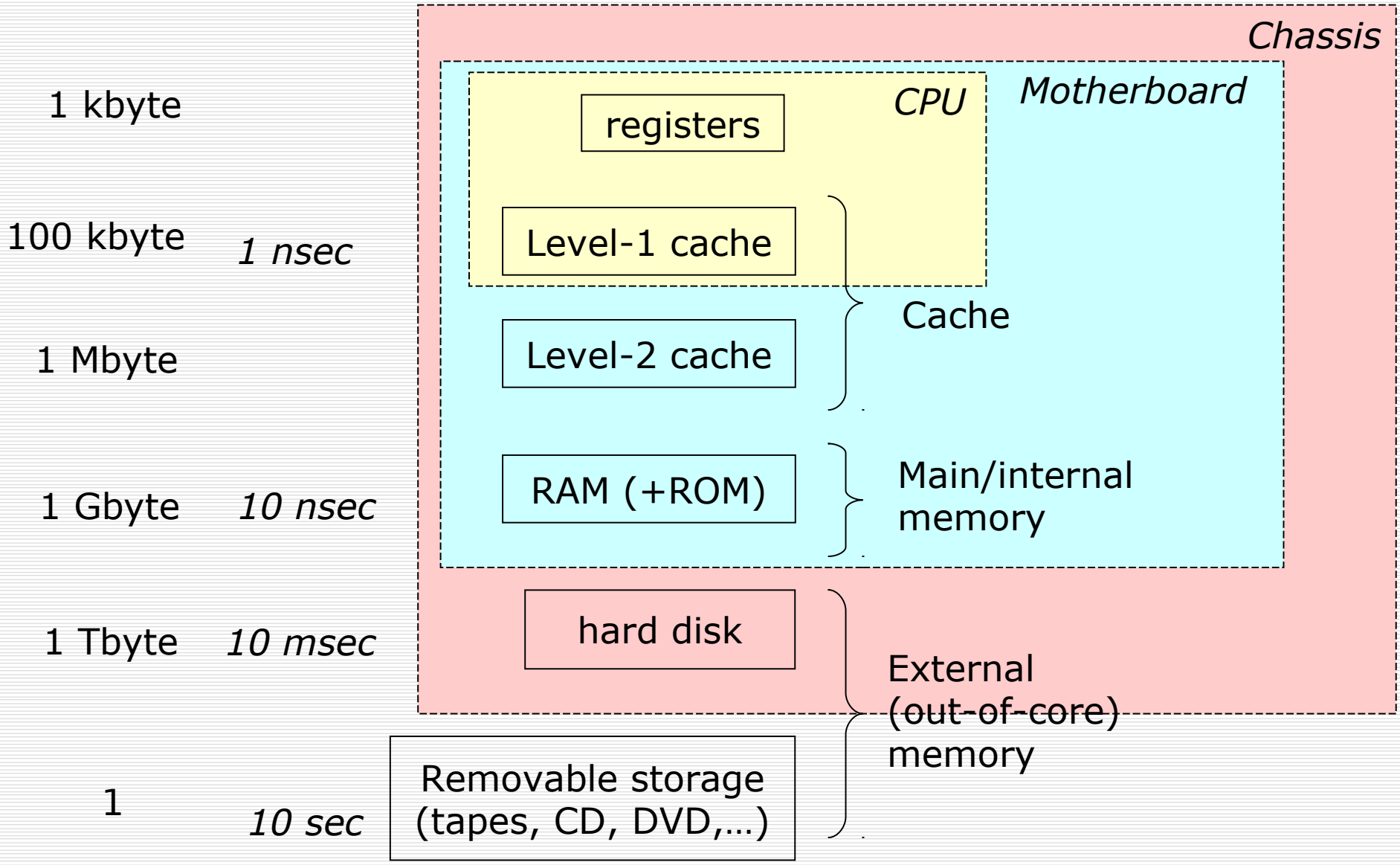
ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ

Overview

- Basic notions
  - Address spaces
  - Memory hierarchy.
  - Locality, spatial/temporal
- Partitioning
  - Allocation
  - Fixed
  - Variable
- Segmentation
- Paging
  - Page tables, TLB
  - Pentium: segmentation+paging
- Process memory
  - Process memory map
  - Heap, malloc/free, garbage collection
  - Mmap, mprotect, sbrk,…
- Tricks
  - Memory-mapped I/O
    - Hi performance I/O (dbs, persistence, etc)
    - Shared libs/code
  - Shared memory (and IPC)
  - Copy-on-write on fork
- Virtual memory
- Swapping
  - overlays
  - Algorithms

Ιεραρχία μνήμης



Γιατί έχουμε cache?

- Locality (τοπικότητα)
  - Spatial (χώρου)
  - Temporal (χρόνου)
- Ορισμός: **spatial locality**

  Αν στο χρόνο t προσπελαστεί η θέση μνήμης x, τότε, με μεγάλη πιθανότητα, στο επόμενο βήμα (t+1) θα προσπελαστεί μια θέση «κοντά» στο x.
- Ορισμός: **temporal locality**

  Αν στο χρόνο t προσπελαστεί η θέση μνήμης x, τότε, με μεγάλη πιθανότητα, θα προσπελαστεί ξανά «κοντά» στο χρόνο t.

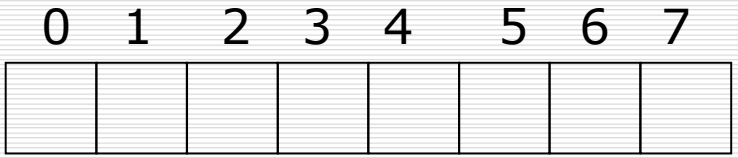
Το πρόβλημα της κατανομής μνήμης (Allocation problem)

- Μνήμη μεγέθους Ν λέξεων
- Ακολουθία αιτήσεων
  - pointer = allocate(n)
  - free(pointer)
  - Σε κάθε χρονική στιγμή ζητείται συνολική μνήμη λιγότερη από Ν.
- Πρόβλημα:
  Η τοποθέτηση των τμημάτων μνήμης έτσι ώστε να εξυπηρετηθούν όλες οι αιτήσεις.

Παράδειγμα

Μνήμη N=8

- P1=allocate(2)
- P2=allocate(2)
- P3=allocate(2)
- P3=allocate(2)
- Free(P2)
- Free(P4)
- P5=allocate(3) ?



Κατακερματισμός

Εφαρμογές

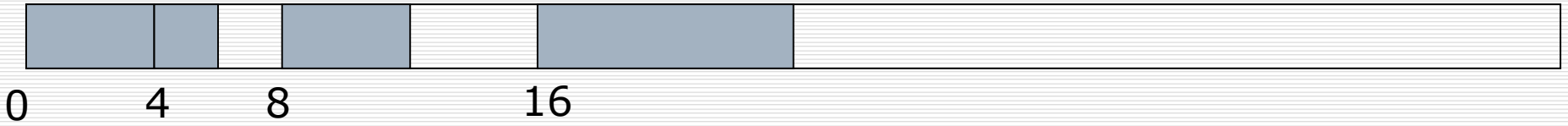
- malloc/free
- Συστήματα αρχείων
  - Δίσκος N bytes = μνήμη N θέσεων
  - Allocate(n) = δημιουργία αρχείου n bytes

Αλγόριθμοι memory allocation

- Στατικό πρόβλημα: η ακολουθία των allocate και free είναι γνωστό από πριν.
- On-line πρόβλημα: οι μελλοντικές αιτήσεις allocate/free δεν είναι γνωστές.
  - Buddy algorithm
  - Best-Fit (BF)
  - First-Fit (FF)
  - Worst-Fit (WF)
  - Next-Fit (NF)
- Heuristic: ελαχιστοποίηση κατακερματισμού
- Κατακερματισμός:
  - Εσωτερικός
  - Εξωτερικός

Buddy algorithm

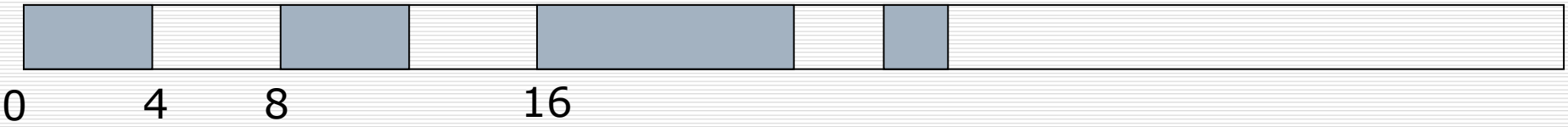
- Όλες οι αιτήσεις allocate(n) στογγυλοποιούνται στην κοντινότερη (προς τα πάνω) δύναμη του 2: $2^{d \log(n)\ e}$.
- Ένα block μεγέθους 2^k τοποθετείται πάντα ώστε η αρχή του να βρίσκεται σε πολλαπλάσιο του 2^k.
- Internal fragmentation.
- Παράδειγμα: allocate 4, 2, 4, 8



Best Fit

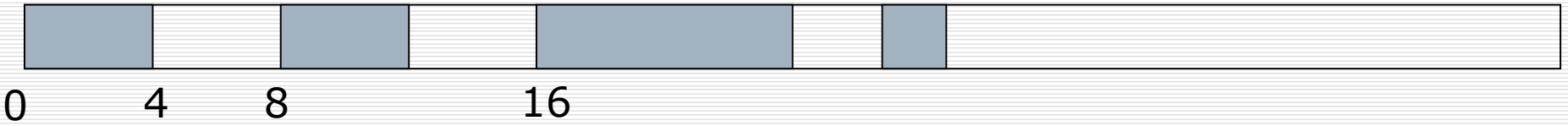
- Η αίτηση ικανοποιείται από (ένα από τα) κενά διαστήματα που αφήνουν τον μικρότερο ανεκμετάλλευτο χώρο.
- External fragmentation.

allocate(3)



First Fit

- Η αίτηση ικανοποιείται από το πρώτο («αριστερότερο») διάστημα που μπορεί να την ικανοποιήσει.
- External fragmentation.



Άλλοι αλγόριθμοι

- Worst Fit: Επιλέγεται το διάστημα που αφήνει το μεγαλύτερο δυνατό υπόλοιπο χώρου.
- Next Fit: Επιλέγεται το διάστημα αμέσως μετά την τελευταία τοποθέτηση.

Σύγκριση αλγορίθμων

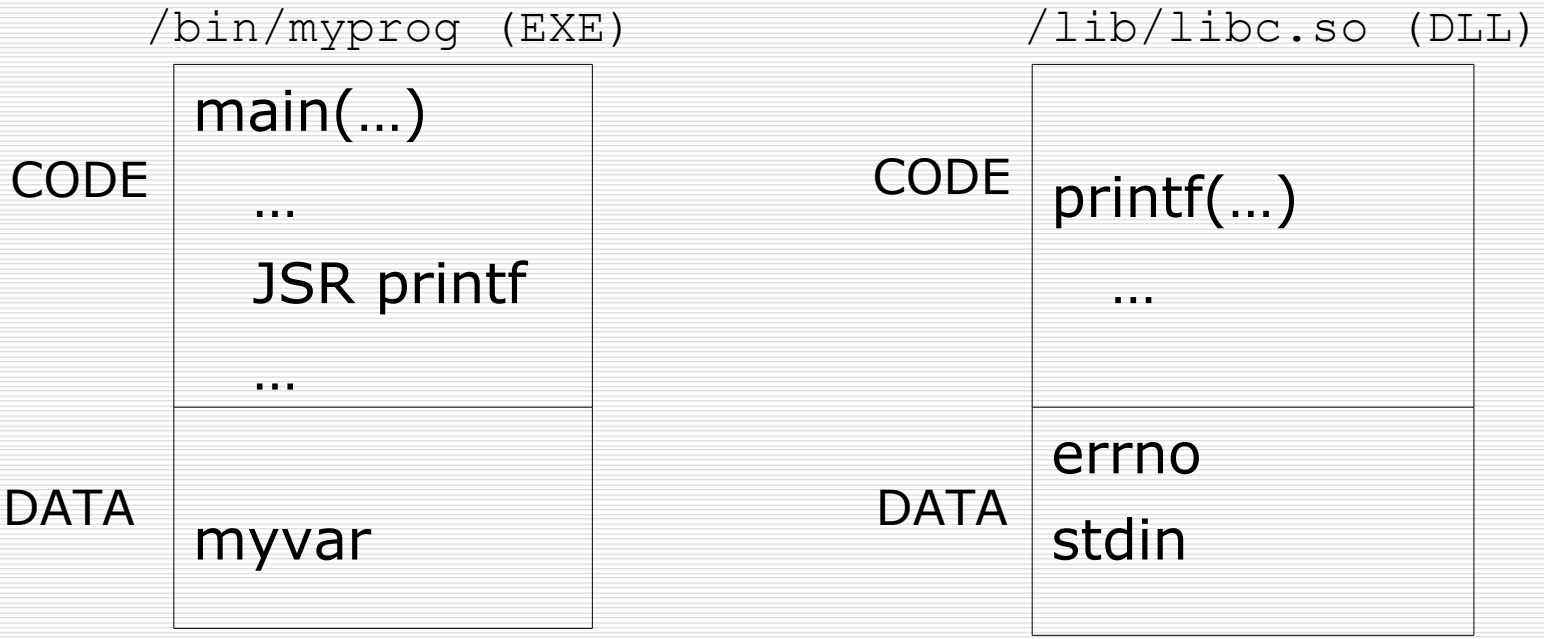
- Με εξομοίωση !!
- Αποτελέσματα:
  - Οι First-Fit και Best-Fit είναι εξίσου καλές.
  - Η buddy έχει μέτρια αποτελέσματα.
  - Οι Next-Fit και Worst-Fit είναι οι χειρότερες (ιδίως η δεύτερη).
- Ιστορικα για τη malloc/free:
  - Στο παρελθόν η malloc/free θεωρούνταν αργή
    - Λόγω κακής υλοποίησης στο παραδοσιακό Unix.
    - Οι προγραμματιστές την απέφευγαν.
  - Σήμερα: άριστες υλοποιήσεις
    - Η υλοποίηση του Dough Lea (best-fit)

Διαχείριση μνήμης

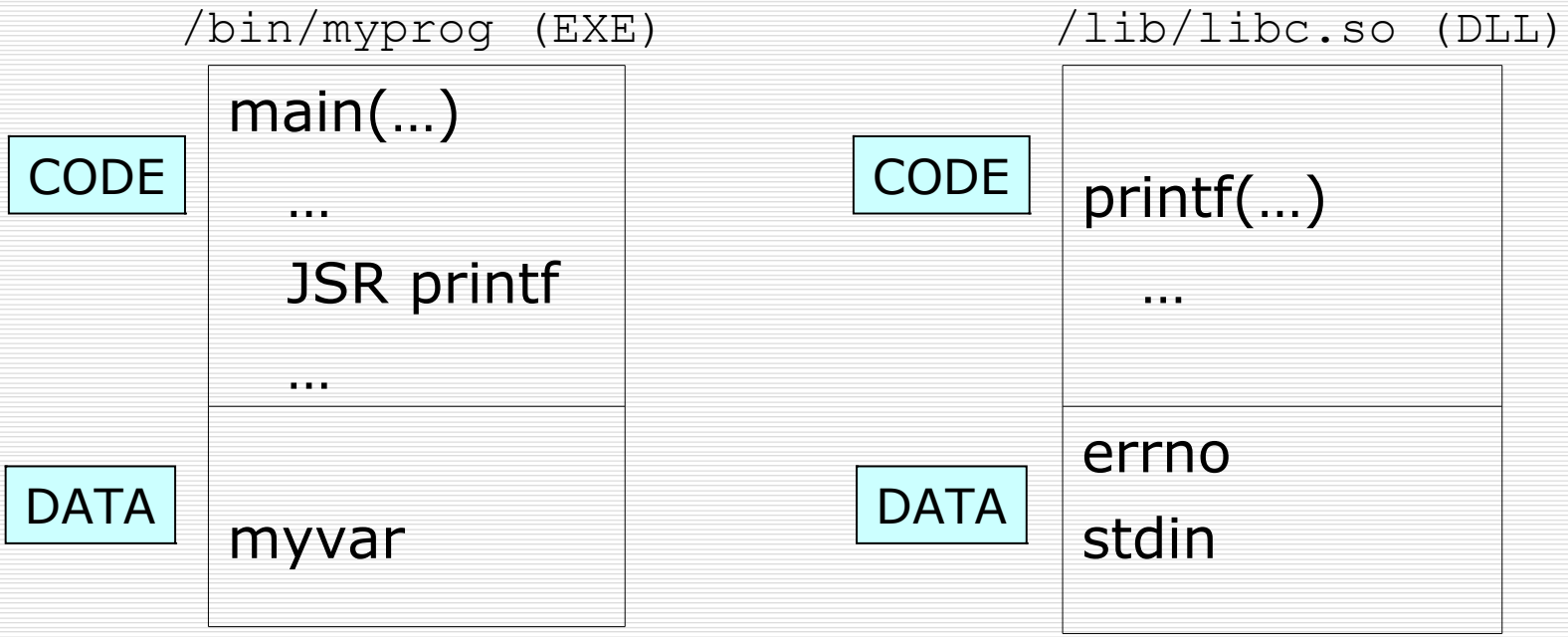
- Οι διεργασίες ζητούν μνήμη από το OS μέσω system calls.
- Η κάθε διεργασία κατανέμει τη μνήμη της σε segments: code, data, stack(s), heap(s)
  - Heaps = allocation problem
- Το OS κατανέμει τη διαθέσιμη φυσική μνήμη στις διεργασίες
  - Allocation problem

Φόρτωση προγραμμάτων

- Στο δίσκο:
  - Προγράμματα και βιβλιοθήκες.
  - Κώδικας.
  - Στατικές μεταβλητές (+ initial values).
  - Πρόβλημα: μετάφραση διευθύνσεων κατά τη φόρτωση.



Φόρτωση στη μνήμη



Dynamic linker (ld.so)

Μνήμη

Address spaces

- Χώρος διευθύνσεων: ένα σύνολο από “λογικές διευθύνσεις”
- Address translation (μετάφραση διευθύνσεων)

  $$T: S \rightarrow S'$$

  Από ένα address space σε άλλο (“λογική σε φυσική” διεύθυνση)
- Στη μνήμη έχουμε address spaces:
  - Διεργασιών (user space)
  - Πυρήνα (kernel space)
  - Μνήμη RAM (physical ή RAM space)
  - Swap space (virtual memory space)
  - Video memory
  - Executable file
  - …

Τρόποι οργάνωσης μνήμης

- Partitioning (Τμηματοποίηση)
- Segmentation (κατάτμηση)
- Paging (σελιδοποίηση)

Partitioning

- Η μνήμη είναι χωρισμένη σε segments
  - Σταθερού μήκους

    A A B B B A

  - Μεταβλητού μήκους

    A B B A A B

- Εσωτερικός κατακερματισμός
- Δεν χρησιμοποιείται πια

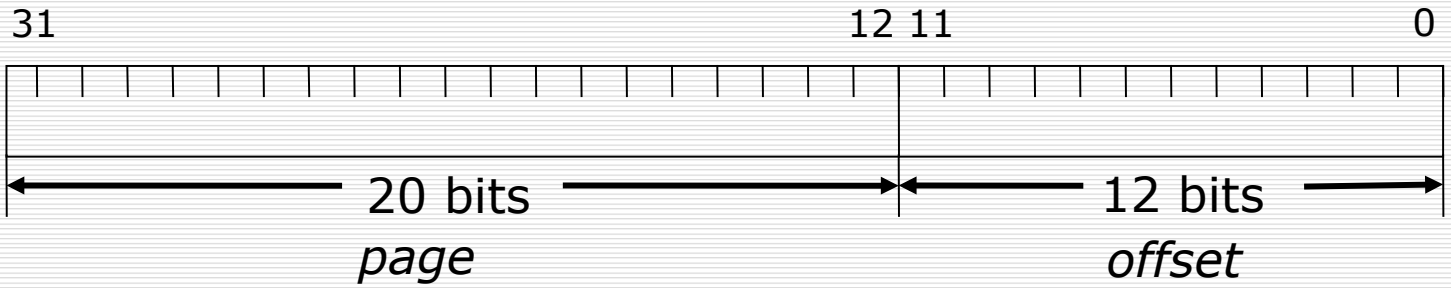
Segmentation

- Κάθε διεργασία στο δικό της address space
- Υποστήριξη από τη CPU
  - Ειδικοί segment registers
  - CS, DS, SS, ES, …
  - Instruction fetch address: CS+IP
  - Data fetch address: DS/ES + effective-address
  - Stack push/pop: SS + SP
  - …
- Εξωτερικός κατακερματισμός
- Intel Pentium

Paging

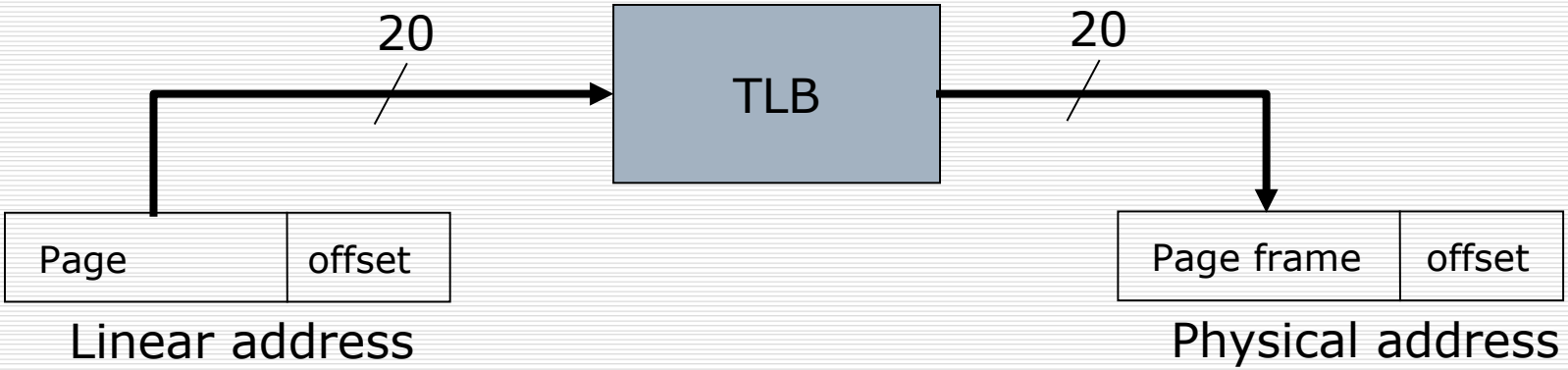
- Η πιο εξελιγμένη δομή
- Δεν έχει καθόλου fragmentation!!!
- Ορισμοί:
  - Page (σελίδα): μια συνεχής περιοχή διευθύνσεων σε κάποιο address space
  - Page frame (πλαίσιο σελίδας): χώρος φυσικής μνήμης μεγέθους μιας σελίδας
  - 4-16 kbytes
  - Pentium: 4 kbytes

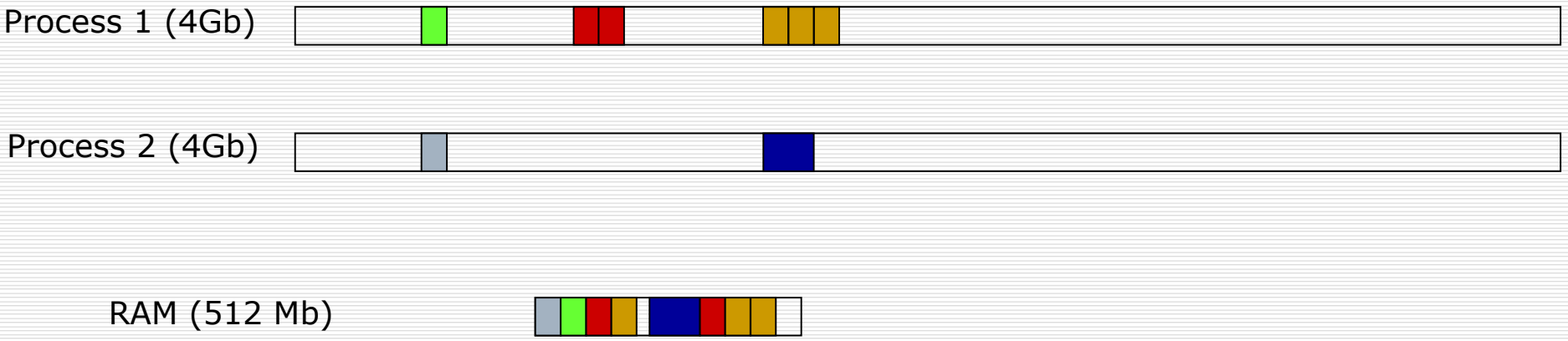
- Address spaces στον Pentium
  - Linear addresses: Διευθύνσεις 32 bit
  - Physical addresses: 32 bit
  - Page size = 2^{12} bytes, 2^{20} pages



Μετάφραση διευθύνσεων

- T[page] -> page frame





Πίνακες σελίδων

- Για κάθε διεργασία: μετάφραση 2^{20} διευθύνσεων σελίδων σε page frames (ή NULL).
- Αφελής υλοποίηση: array

page	Page frame (20 bit)	Valid (1 bit)	Access-ed (1 bit)	Dirty (1 bit)	R (1 bit)	W (1 bit)
0						
	…					
2^{20}-1						

TLB: μια cache για τον *εκάστοτε* πίνακα σελίδων

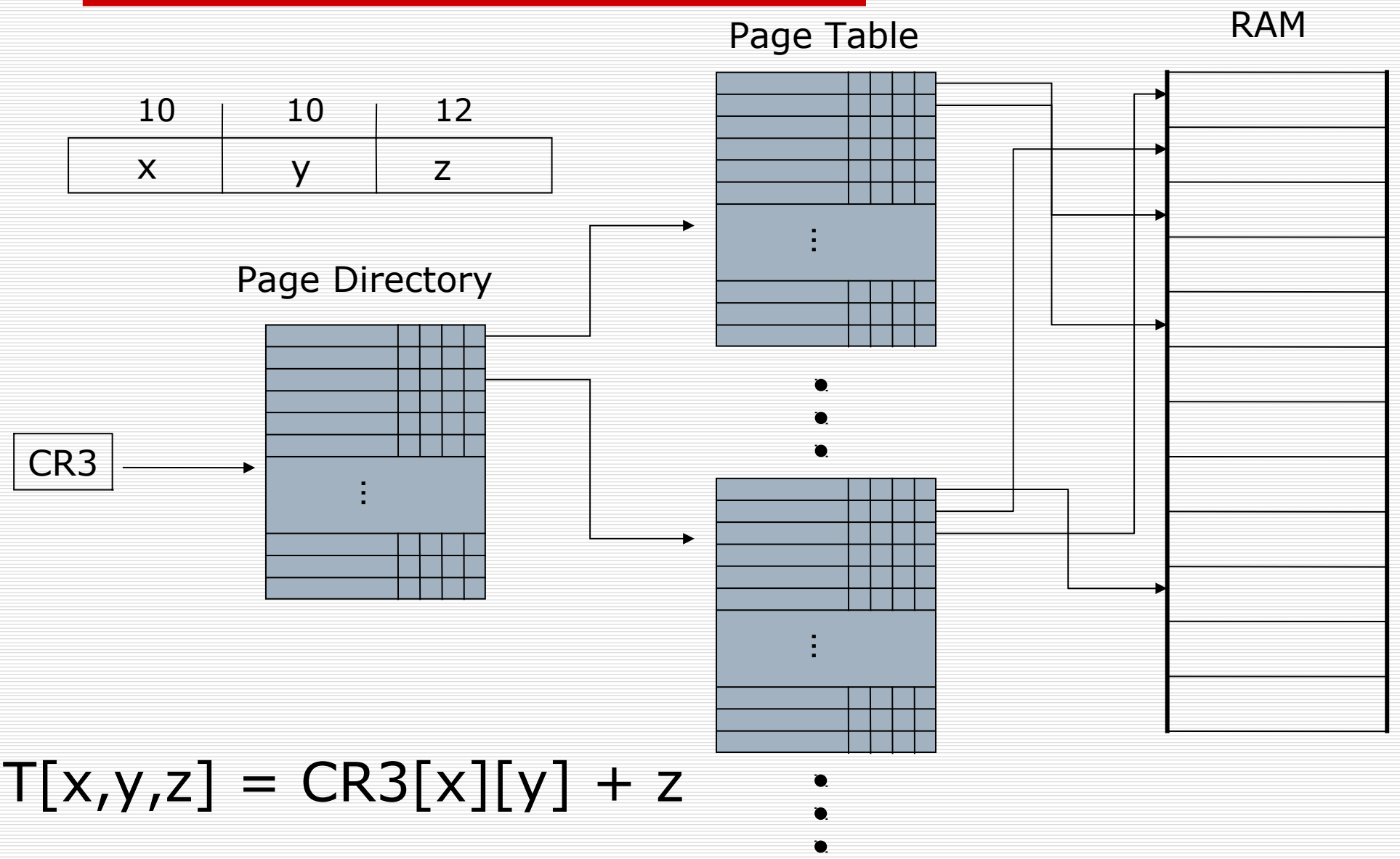
Πίνακες σελίδων στον Pentium

- Πρόβλημα με αφελή υλοποίηση: μέγεθος

$$2^{20} \text{ £ } 3 \text{ bytes} = 3\text{Mbytes}$$

- Ανά διεργασία !!
- Παρατήρηση: η πλειοψηφία των εγγραφών είναι NULL.
- Λύση: ιεραρχικοί πίνακες σελίδων

2-level page tables



T[x,y,z] = CR3[x][y] + z

Δομή του page table

- Επίπεδο 1: Σελίδα από PDEs (Page Directory Entries)
- Επίπεδο 2: Σελίδες από PTEs (Page Table Entries)

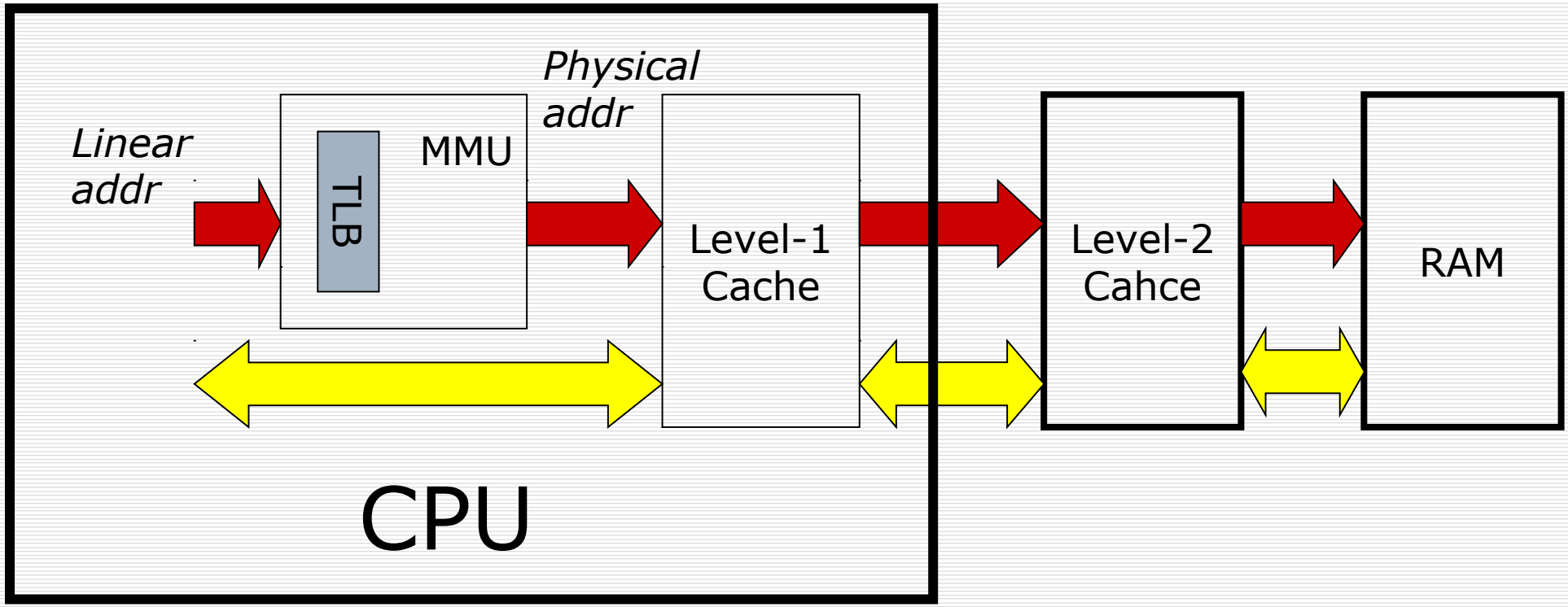
PTE	Page frame (20 bit)	Valid (1 bit)	Accessed (1 bit)	Dirty (1 bit)	R (1 bit)	W (1 bit)	Other

- **Valid:** 1 αν η σελίδα είναι νόμιμη, 0 αλλιώς.
- **Accessed:** Γίνεται 1 σε κάθε προσπέλαση στη σελίδα.
- **Dirty:** Γίνεται 1 σε κάθε εγγραφή.
- **R:** 1 αν επιτρέπεται η ανάγνωση, 0 αλλιώς.
- **W:** 1 αν επιτρέπεται η εγγραφή, 0 αλλιώς.
- ***Other***: Το OS τα χρησιμοποιεί για τους μηχανισμούς του.

Σφάλματα μνήμης

- Kernel: γίνεται trap αν
  - Προσπελαστεί σελίδα με Valid=0.
  - Γίνει ανάγνωση σε σελίδα με R=0.
  - Γίνει εγγραφή σε σελίδα με W=0.
- Αν το σφάλμα είναι «αποδεκτό» (πχ. σελίδα μνήμης στο δίσκο –virtual memory), τότε το ΛΣ χειρίζεται κατάλληλα.
- Αν το σφάλμα είναι της διεργασίας, στέλνεται SIGSEGV.

Υλοποίηση


Linear
addr
TLB
MMU
Physical
addr
Level-1
Cache
Level-2
Cahce
RAM
CPU

System calls

- mmap(start, len, prot, flags, fd, offset)
  - **start, len**: θέση και μέγεθος νέας περιοχής μνήμης (αν start==NULL, το ΛΣ διαλέγει τη θέση).
  - **prot**: protection,
    PROT_EXEC | PROT_READ | PROT_WRITE | PROT_NONE
  - **flags**:
    MAP_FIXED: Χρησιμοποίησε τη θέση start
    MAP_SHARED: Κοινή σελίδα
    MAP_PRIVATE: Ιδιωτική σελίδα
    MAP_ANONYMOUS: Αγνοεί fd, len (όμοιο με fd=/dev/zero)
  - **fd, offset**: αρχείο και θέση μέσα του για απεικόνιση

System calls

- mremap(…)
  Μεταβάλλει το μέγεθος ενός mapping
- munmap(start,len):
  Απελευθερώνει τη μνήμη
- mprotect(start,len,prot):
  Μεταβάλλει την προστασία μνήμης
- msync(start,len,flags):
  Στέλνει «βρώμικες» σελίδες στο δίσκο

Μηχανισμοί μνήμης

- Υλοποιούνται με τα flags του page table. Valid, Accessed, Dirty, R, W

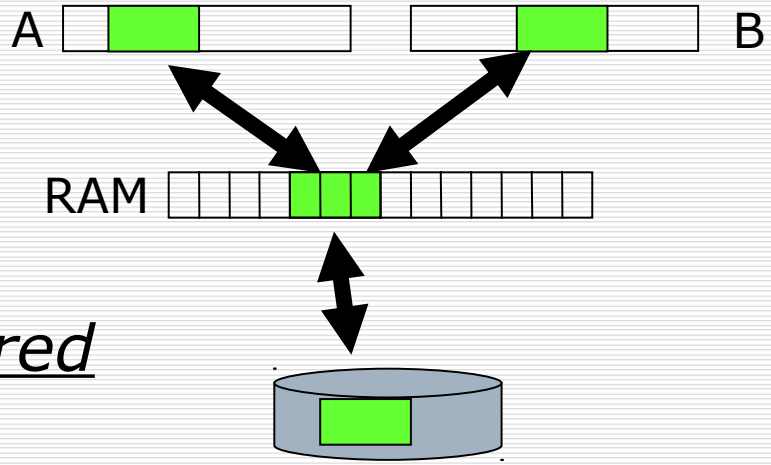
- Memory-mapped I/O
  - mprotect, msync
  - Shared vs. private
  - Shared memory
  - Shared libraries/executables
  - Copy-on-write
  - Γρήγορη υλοποίηση fork
- Virtual memory

Υλοποίηση mmap, mprotect, msync

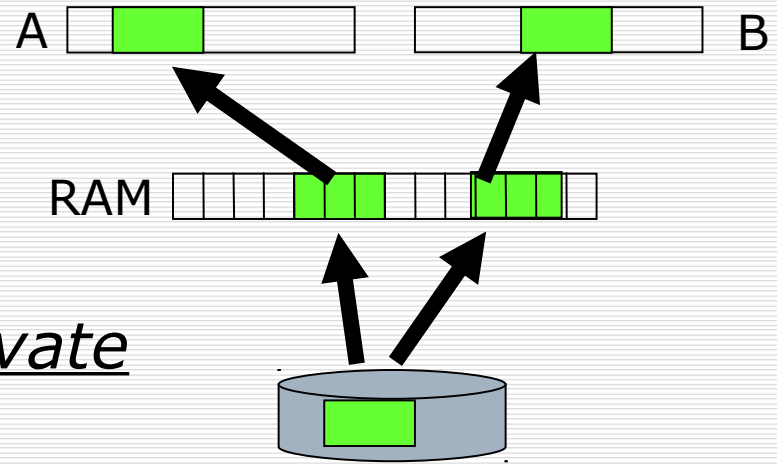
- mmap(file, len) -> addr
  - Επιλογή addr μήκους len στο address space της διεργασίας ➔ allocation problem !
  - Δεν εκτελείται άμεσα I/O δίσκου!!
  - Mode bits: Valid = false , Accessed = false, Dirty = false, R/W αναλόγως.
  - I/O εκτελείται όταν γίνει προσπέλαση σε :Valid σελίδα.
- mprotect:
  - Μεταβάλλει τα R/W μόνον
- msync:
  - Γράφει τις Dirty σελίδες στο δίσκο.
  - Εκτελείται και κατά το munmap(…).

Shared vs. private mappings

- Αν δύο διεργασίες κάνουν map το ίδιο αρχείο?
  - MAP_SHARED/MAP_PRIVATE
- Shared: Read/Write στο δίσκο, shared page frames
- Private: Read-only (διαβάζεται μόνο στην πρώτη προσπέλαση – αρχικοποίηση), χωριστά page frames.



Shared



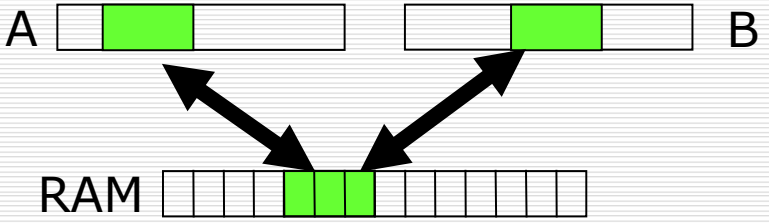
Private

Shared memory

- Η mmap χωρίς αρχείο = MAP_ANONYMOUS
- Χρήσεις:
  - Private maps: η διεργασία παίρνει επιπλέον μνήμη (κυρίως για stack ή heap).
  - Shared maps: ➔ Shared memory για επικοινωνία
- Εναλλακτικά στο Unix: map του αρχείου `/dev/zero`

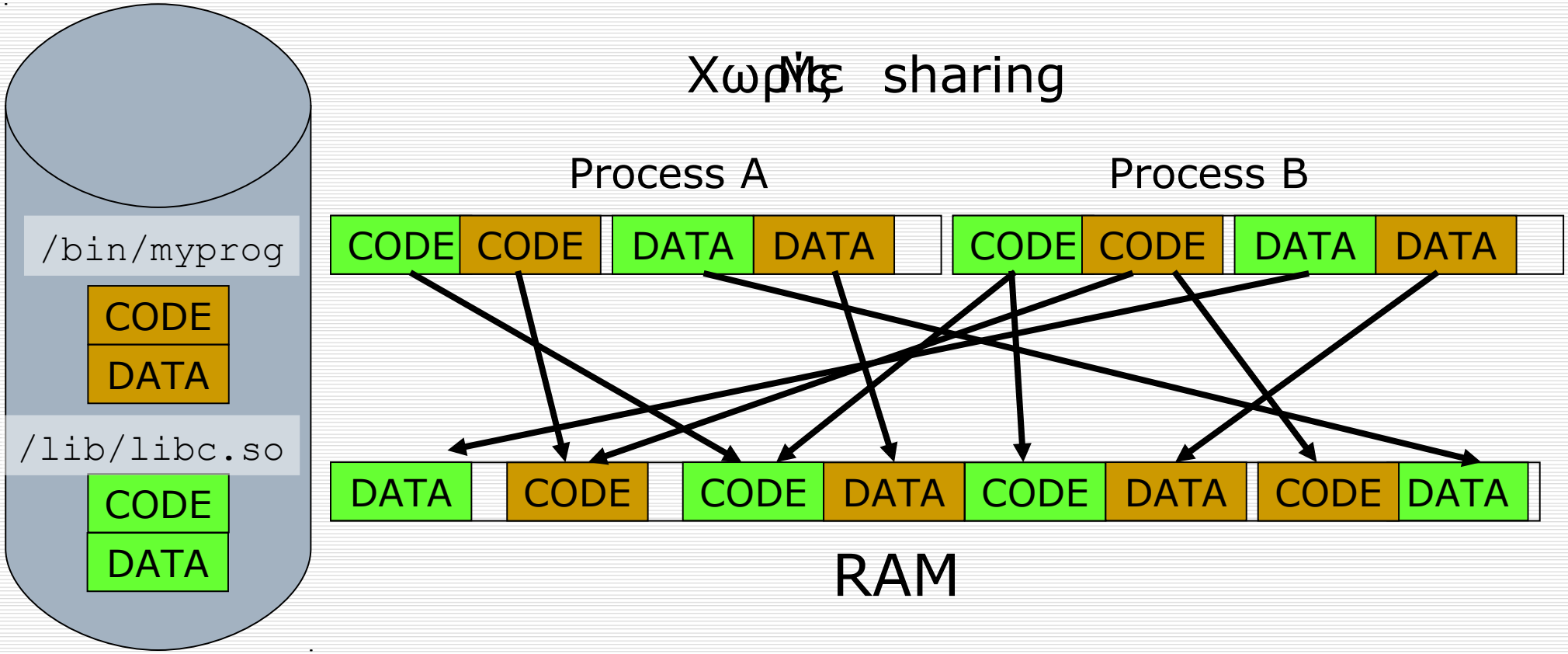
Private anonymous mappings

Shared memory



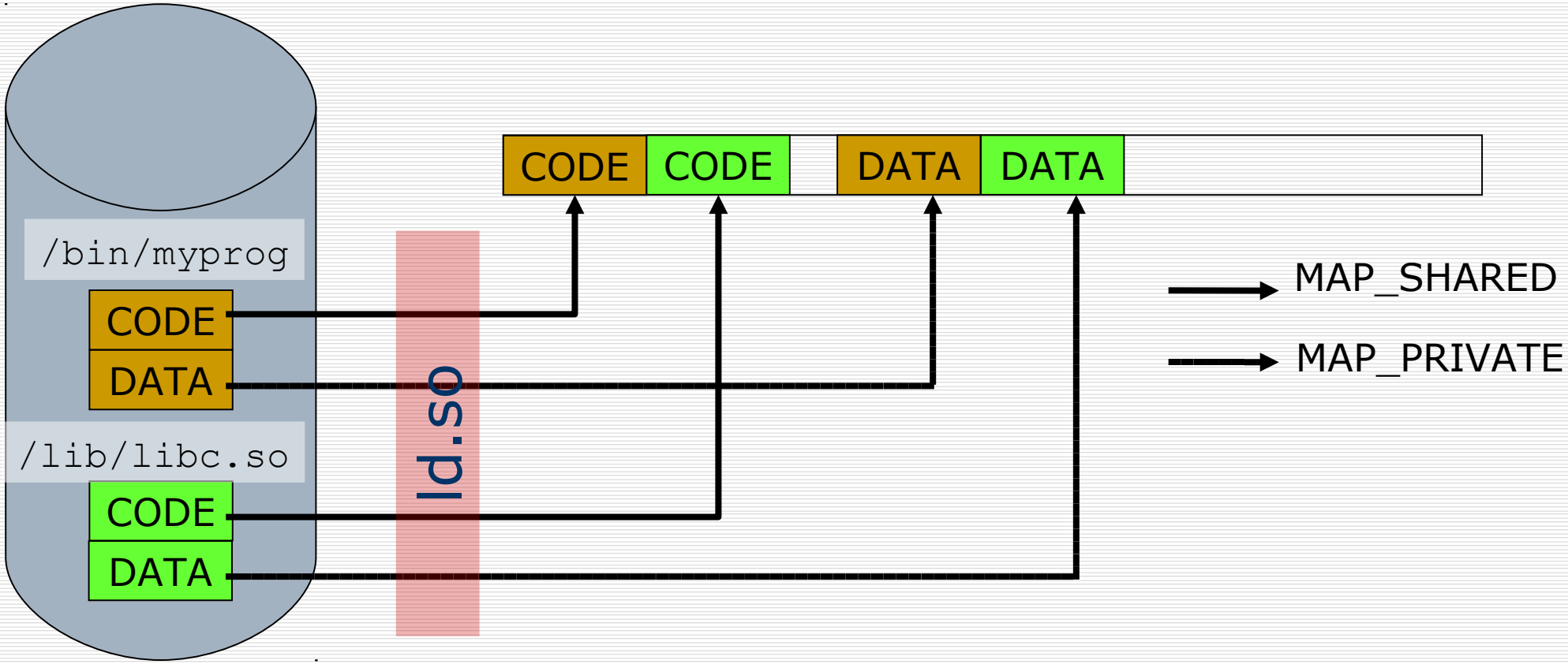
Εφαρμογή: shared binaries

- Φόρτωση binaries
- Κέρδος μεγάλο κυρίως για βιβλιοθήκες



Υλοποίηση shared binaries

- Binary files: χωριστά code και data segments
- Dynamic linker (ld.so): κάνει τα κατάλληλα mappings.



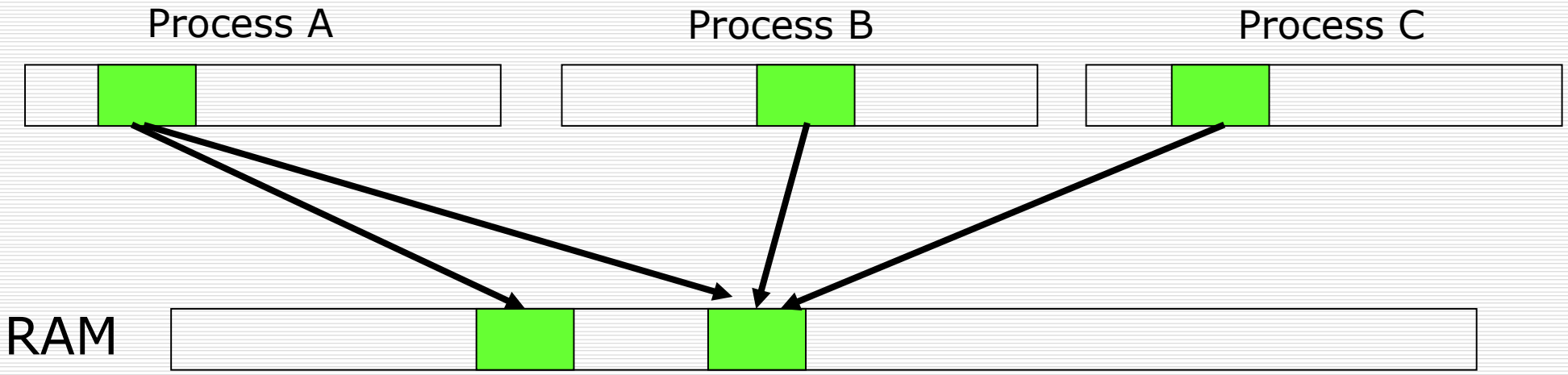
Περιορισμοί υλοποίησης

- Μειονέκτημα shared binaries: τι γίνεται αν αλλάξει το αρχείο (πχ. το myprog)?
  - Απ: ΧΑΟΣ!!
  - Αρα: ένα executable που είναι mmaped δε θα πρέπει να μπορεί να αλλάξει.

- Γενική λύση μέσω του copy-on-write

Copy-on-write

- Lazy copy (οκνηρή αντιγραφή)
- Αντιγραφή σελίδας/ων μνήμης από ένα address space σε άλλο
  - Αρχικά η αντιγραφή γίνεται με sharing
  - Τη στιγμή που γίνεται απόπειρα εγγραφής, δημιουργείται αντίγραφο.



Υλοποίηση copy-on-write

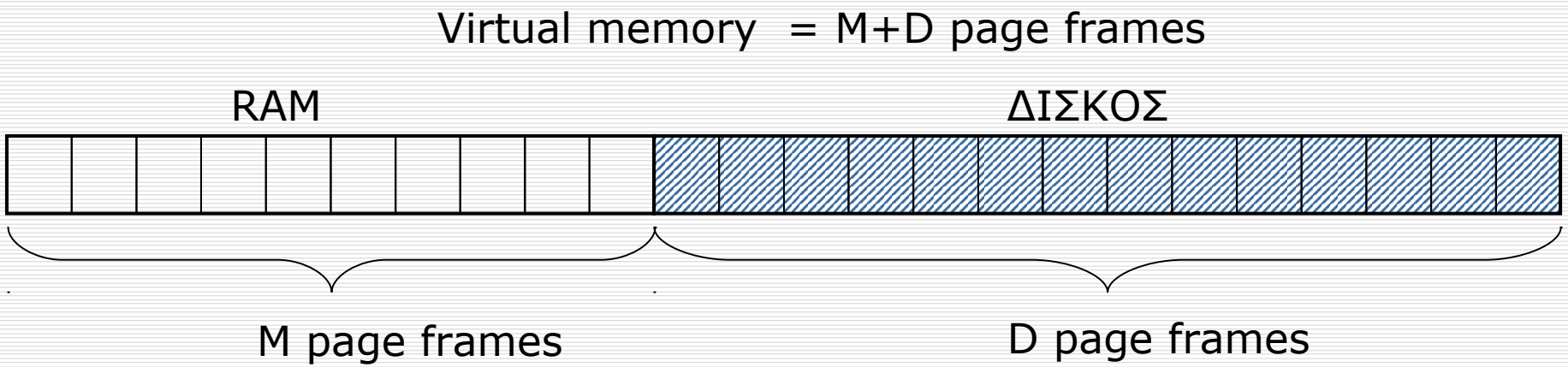
- Αρχικά:
  - Οι σελίδες μαρκάρονται :W.
  - Χρησιμοποιείται ένα από τα διαθέσιμα bits του PTE για να μαρκάρει τη σελίδα ως copy-on-write.
  - Για κάθε shared σελίδα αποθηκεύεται ο αριθμός των references (αναφορών).
- Σε απόπειρα εγγραφής:
  - Η σελίδα αντιγράφεται
  - Μειώνεται ο αριθμός των references
    - Αν τα references γίνουν 1, η σελίδα ξεμαρκάρεται από copy-on-write.

Εφαρμογές copy-on-write

- Shared binaries
  - Ο κώδικας φορτώνεται και αυτός MAP_PRIVATE (όπως το DATA segment).
  - Δεν υπάρχει δαπάνη μνήμης επειδή η αντιγραφή γίνεται copy-on-write.
- Fork
  - Η αντιγραφή της μνήμης του πατέρα στο παιδί γίνεται copy-on-write.

Virtual memory

- Επέκταση μνήμης στο δίσκο.
- M+D = η συνολική μνήμη που μπορεί να διατεθεί στις διεργασίες



Swapping

- Ορισμός: Page fault
  Μια προσπέλαση όπου η αντίστοιχη σελίδα δεν βρίσκεται στην RAM.
- Κόστος page fault ¼ 10msec !!!!

- Replacement (αντικατάσταση)
  - Όταν δεν υπάρχει αχρησιμοποίητο page frame, κάποιο από τα γεμάτα *επιλέγεται* για χρήση.
  - Σκοπός της επιλογής αντικατάστασης: ελαχιστοποίηση των page faults (μελλοντικά).

Page replacement problem

- Έχουμε
  - έναν αριθμό M από page frames.
  - Μια ακολουθία από προσπελάσεις

M=4 

Προσπελάσεις: 0 3 5 6 4 6 3 9 5 7 8 3 4 5 3 5 7 3 0 4 3 0 2 3 5 ...

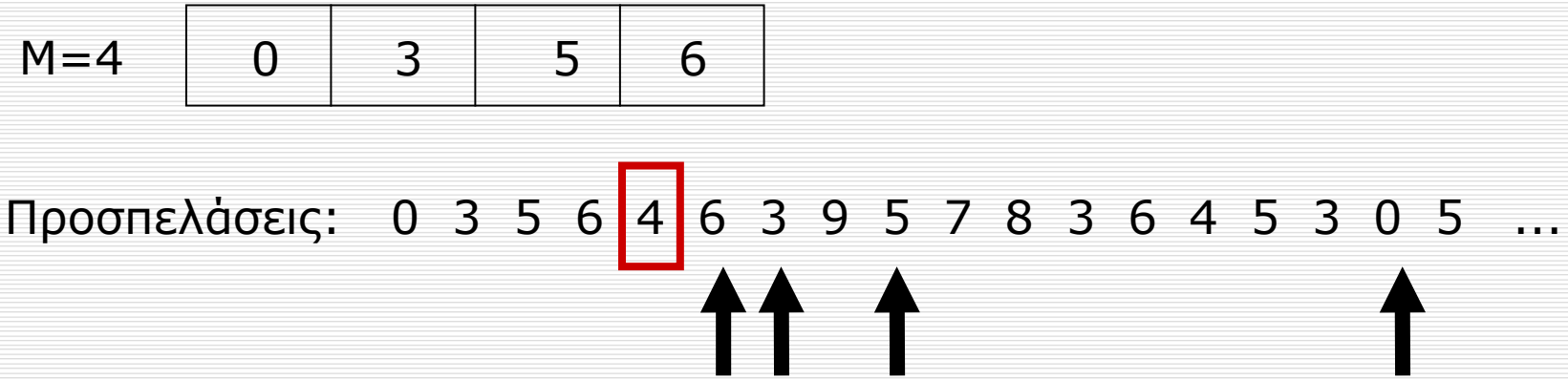
- Αν η προσπελαυνόμενη σελίδα δεν βρίσκεται σε κάποιο πλαίσιο, πρέπει να τοποθετηθεί.
- Αν δεν υπάρχει κενό πλαίσιο, πρέπει να γίνει αντικατάσταση
- Πρόβλημα: επιλογή της σελίδας προς αντικατάσταση

VM και Locality

- VM: εκμεταλεύεται και spatial και temporal locality.
- Ορισμός:
  Working set μεγέθους n: το σύνολο των n πιο πρόσφατων σελίδων που χρησιμοποίησε η διεργασία.
- Λόγω locality, το working set δεν μεταβάλλεται συχνά.

Βέλτιστη αντικατάσταση

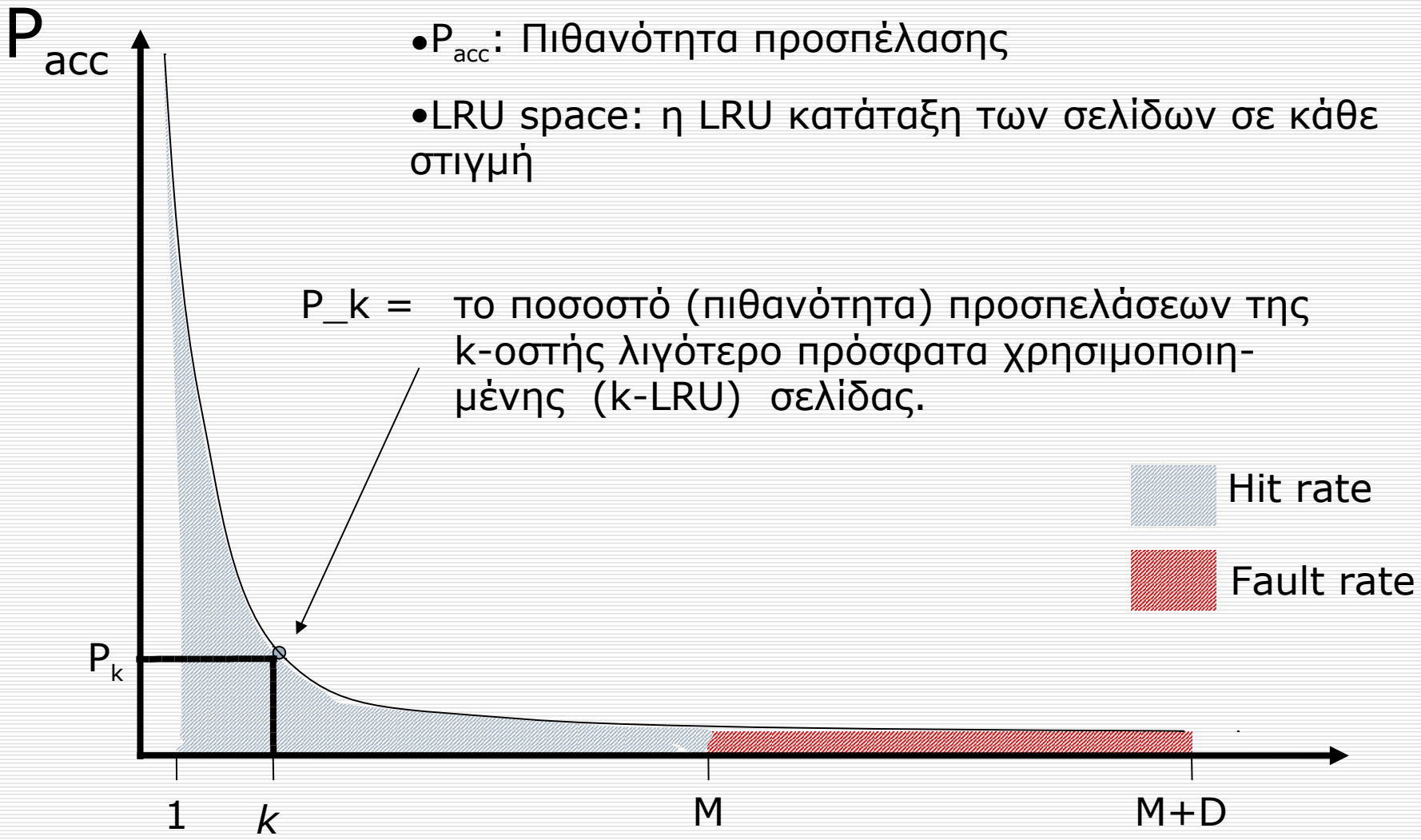
- OPT:
  Αντικαθίσταται η σελίδα που θα προσπελαστεί αργότερα στο μέλλον
- Παράδειγμα



Βασικοί αλγόριθμοι

- FIFO: Αντικαθιστά την παλιότερη σελίδα.
  - Συχνά αυτή είναι κακή επιλογή
  - Εύκολη υλοποίηση
- RANDOM: Επιλέγεται μια τυχαία σελίδα για αντικατάσταση.
  - Δεν είναι τόσο κακό όσο ακούγεται !!
  - Πολύ εύκολη υλοποίηση
- LFU: Least Frequently Used
  - Συχνά κακή επιλογή
  - Δύσκολη υλοποίηση
- **LRU: Least Recently Used**
  - Βέλτιστη απόδοση
  - Δύσκολη υλοποίηση
  - Προσεγγίσεις: CLOCK, NFU (ageing)

Κατανομή LRU

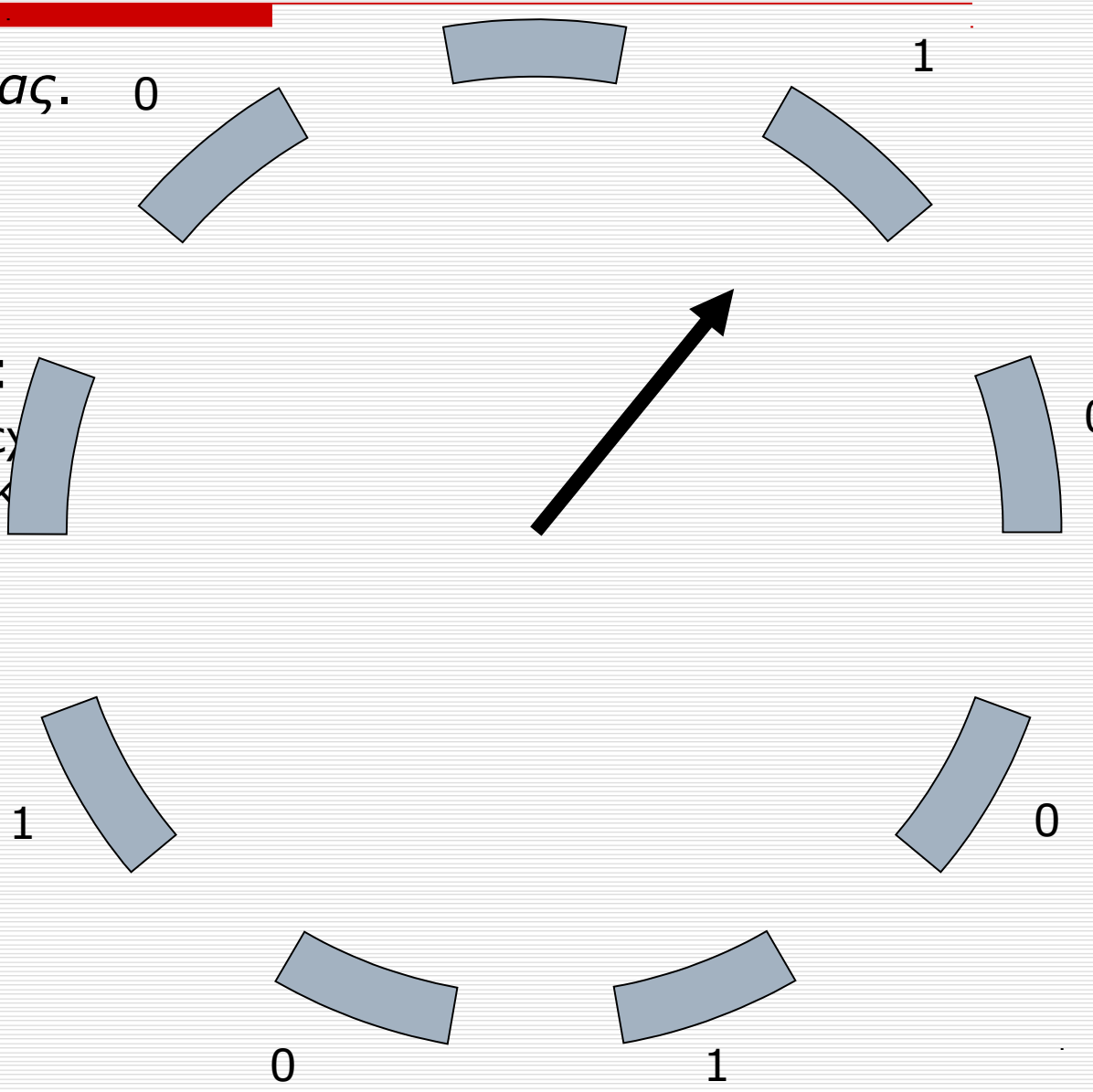


Υλοποίηση LRU

- Γενικά δύσκολη, λόγω
  - Πρέπει κάθε προσπέλαση μνήμης να σημειώνεται
  - Σε αντίθεση με πχ. FIFO
- Προσεγγίσεις
  - Χρησιμοποιείται το Accessed bit των PTE.
  - Το bit μηδενίζεται περιοδικά.
  - Δύο μέθοδοι προσέγγισης
    - Clock
    - Aging

Αλγόριθμος clock (ρολογιού)

- A.k.a. *δεύτερης ευκαιρίας*.
- Οι resident σελίδες διατάσσονται σε κύκλο.
- Τη στιγμή της επιλογής:
  - Αν η τρέχουσα σελίδα έχ
    Accessed=1, γίνεται 0 κ
    ο δείκτης προχωρεί.
  - Αλλιώς επιλέγεται η τρέχουσα σελίδα.



Ageing (παλαίωση)

- Κάθε σελίδα p έχει ένα μετρητή C[p] των k bits.
- Σε κάθε κύκλο
  - C[p] := Accessed << (k-1) + C[p] >> 1
  - ή
  - C[p] = $2^{(k-1)}$ £ Accessed + C[p]/2
- Επιλέγεται η σελίδα με το μικρότερο C[p].

A		Πριν					Μετά			
0		1	1	0	1		0	1	1	0
1		0	0	1	1		1	0	0	1
1		1	0	1	0		1	1	0	1