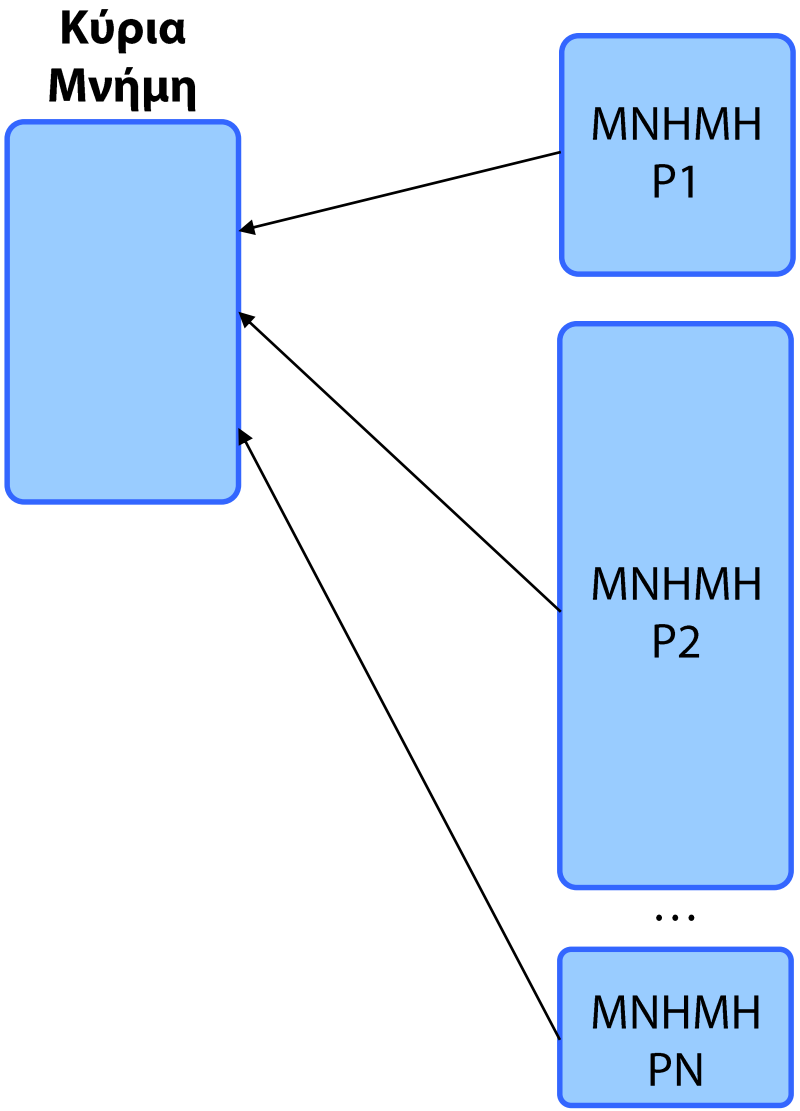
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχ. και Μηχανικών Υπολογιστών
Εργαστήριο Υπολογιστικών Συστημάτων

Διαχείριση Μνήμης – Εικονική Μνήμη (επανάληψη)

Λειτουργικά Συστήματα Υπολογιστών

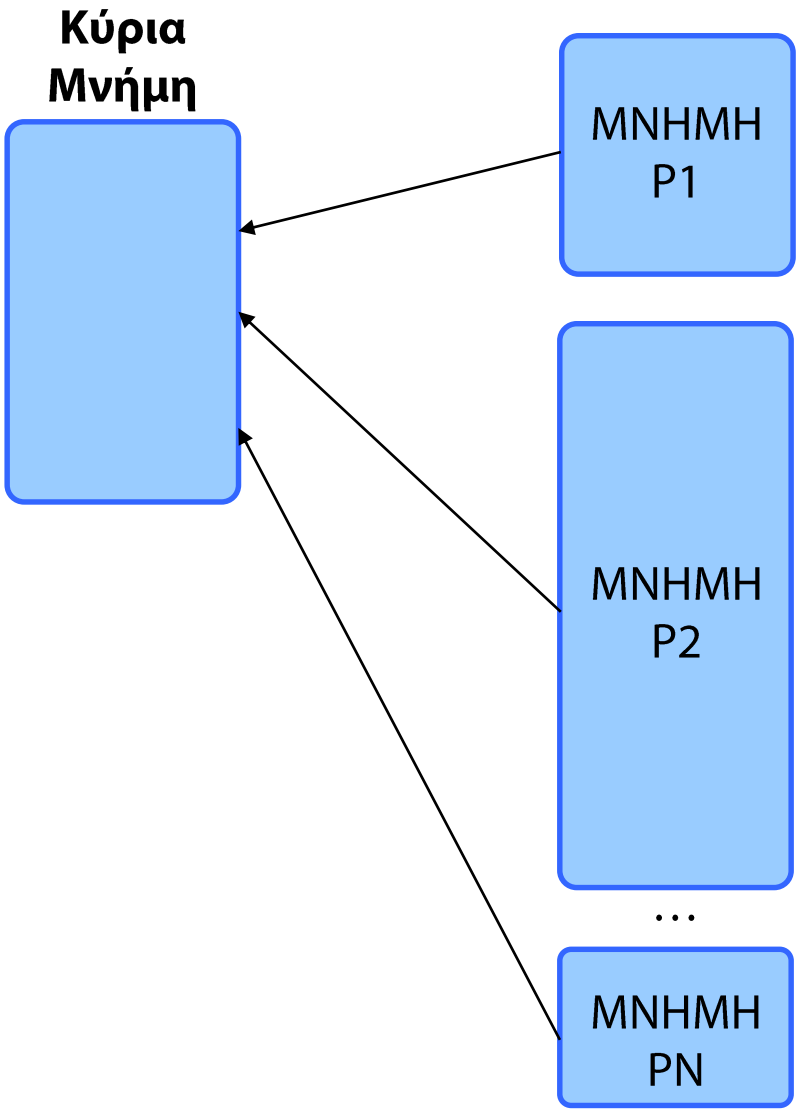
6ο Εξάμηνο, 2019-2020

Ποιο είναι το πρόβλημα;



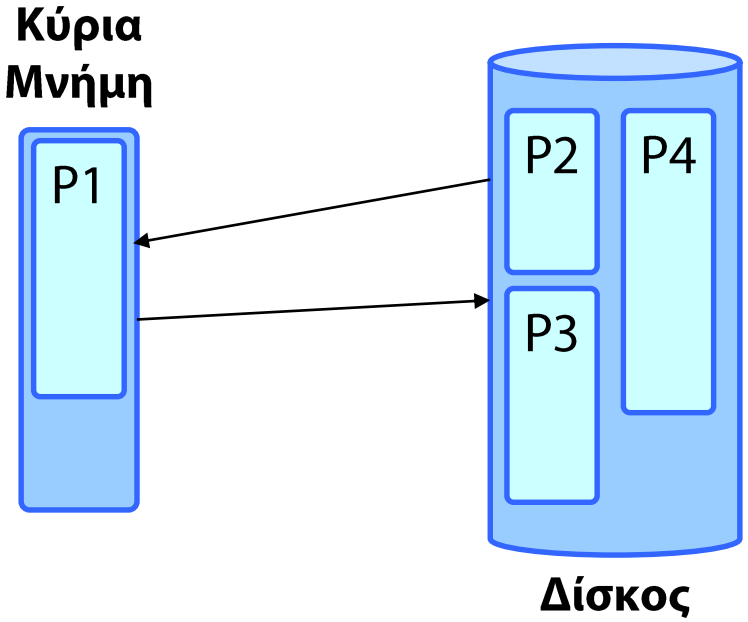
- Ν διεργασίες χρειάζονται πρόσβαση στη φυσική μνήμη (Κύρια Μνήμη).
- Η συνολική απαίτηση για μνήμη μπορεί να είναι μεγαλύτερη από τη διαθέσιμη μνήμη
- Μία διεργασία μπορεί να απαιτεί περισσότερη μνήμη από τη διαθεσιμη
- Οι διεργασίες εισέρχονται και εξέρχονται δυναμικά στο σύστημα.
- Οι διεργασίες αλλάζουν τις απαιτήσεις σου σε μνήμη κατά τη διάρκεια της εκτέλεσης (στοίβα, σωρός)

Επιθυμητά χαρακτηριστικά λύσης



- Καλή χρήση της μνήμης (πολύτιμος πόρος)
- Υψηλή επίδοση
- Απλότητα (χαμηλό κόστος, ευκολία υλοποίησης/ προγραμματισμού, κλπ)

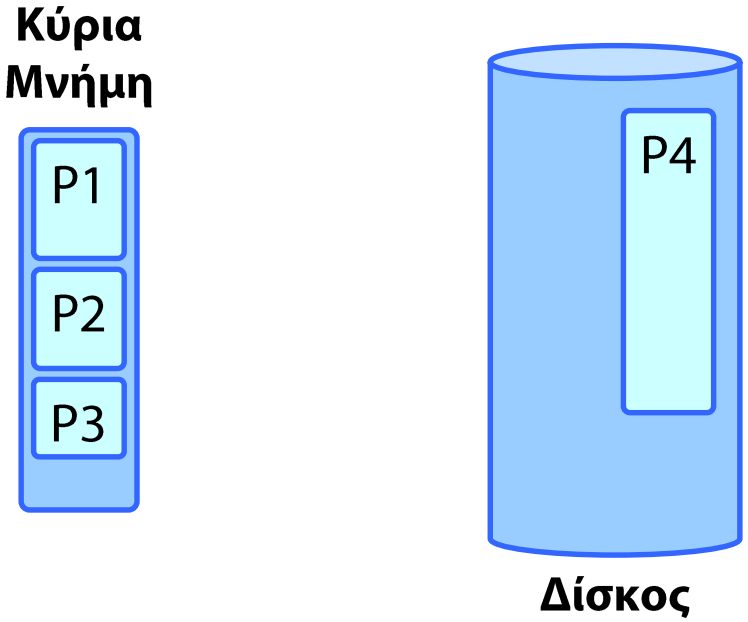
Λύση #0: Μία διεργασία στη μνήμη



Σε κάθε εναλλαγή διεργασιών, αντικαθιστά η προς εκτέλεση διεργασια την υπό εκτέλεση

- Καλή χρήση της μνήμης
  - Υποχρησιμοποίηση για «μικρές» διεργασίες
  - Αδυναμία εκτέλεσης για «μεγάλες» διεργασίες
- Υψηλή επίδοση
  - Συνεχής χρήση δίσκου σε κάθε context switch
- Απλότητα
  - Ελάχιστη υποστήριξη υλικού
  - Διευθύνσεις στο χρόνο μεταγλώττισης

Λύση #1: Συνεχής ανάθεση

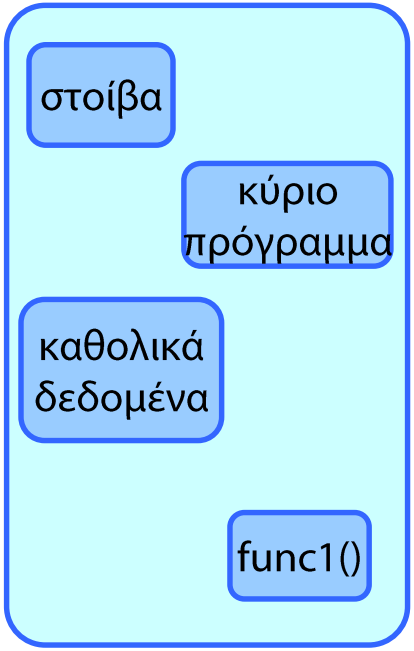


Βλ. διαφάνειες 17-20 και 26-53 από την παρουσίαση «Διαχειριση Κύριας Μνήμης»)

- Καλή χρήση της μνήμης
  - Κατακερματισμός
  - Δύσκολη δυναμική παραχώρηση μνήμης
  - Αδυναμία εκτέλεσης για «μεγάλες» διεργασίες
- Υψηλή επίδοση
  - Γρήγορο για «λίγες» διεργασίες που χωρούν στη μνήμη
  - Σημαντική χρήση δίσκου για «πολλές» διεργασιες
- Απλότητα
  - Μικρή υποστήριξη υλικού (καταχωρητές βάση – όριο)
  - Διευθύνσεις στο χρόνο εκτέλεσης

Λύση #2: Κατάτμηση (segmentation)

- Μικρή βελτίωση του σχήματος συνεχούς ανάθεσης
- Η μνήμη της διεργασίας διαχωρίζεται σε τμήματα (segments)
- Αμβλύνει, αλλά δεν λύνει τα προβλήματα της συνεχούς ανάθεσης



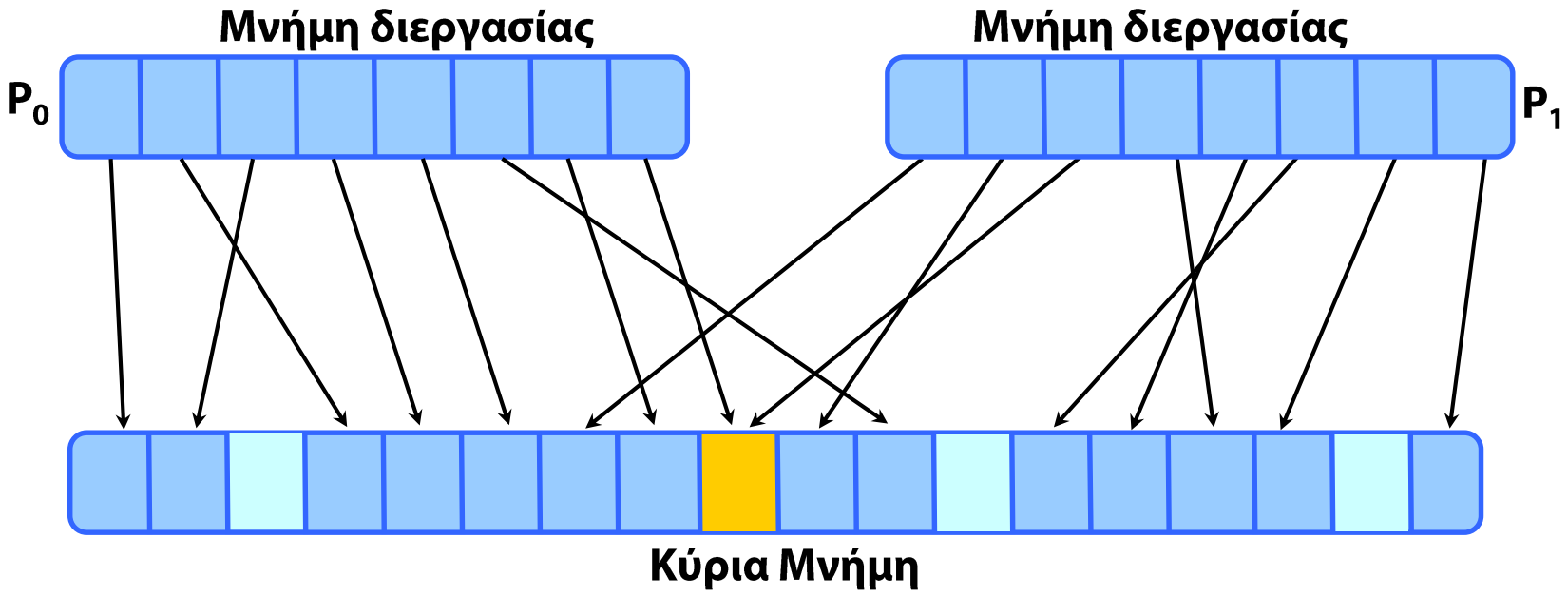
Αναζητώντας μια καλή λύση

- Πρέπει να βελτιώσουμε την αξιοποίηση της μνήμης
  - Αν υπάρχει διαθέσιμη μνήμη, να μπορούμε να τη χρησιμοποιήσουμε
  - Συνύπαρξη πολλών διεργασιών που εισέρχονται / εξέρχονται στο σύστημα
  - Δυναμική παραχώρηση μνήμης ανά διεργασία
  - Υποστήριξη «μεγάλων» διεργασιών
- Η επίδοση θα πρέπει να είναι υψηλή στην κοινή περίπτωση και αποδεκτή σε «δύσκολες» καταστάσεις
- Έχουμε τη δυνατότητα να υλοποιήσουμε πιο σύνθετες λύσεις με υποστήριξη από το υλικό, προσπαθώντας να κρύψουμε την πολυπλοκότητα από τον τελικό χρήστη (προγραμματιστή).

Αναζητώντας μια καλή λύση (συν.)

- **Παρατήρηση:** Πόσο αναγκαίο είναι μια διεργασία να είναι ολόκληρη στη μνήμη;
  - Μια διεργασία μπορεί να μην χρειαστεί ποτέ τη μνήμη που έχει ζητήσει
    - Να μην εκτελέσει πότε μια συνάρτηση που υπάρχει στον κώδικά της
    - Να μην αγγίξει ποτέ δεδομένα που έχει δεσμεύσει
  - Μια διεργασία δουλεύει σε ένα υποσύνολο των δεδομένων της εντός χρονικού παραθύρου (βλ. σύνολο εργασίας / working set, διαφάνειες 168... παρουσίαση Εικονική Μνήμη 2/2)
  - Επίσης θυμηθείτε: fork() -> execv
- **Ιδέα:** Να μοιράσουμε τη μνήμη κάθε διεργασίας σε σταθερά κομμάτια (θα τα λέμε σελίδες – pages) και θα τοποθετούμε στη μνήμη κάθε σελίδα ανεξάρτητα (θα αναθέτουμε σελίδες σε πλαίσια-frames μνήμης)

Λύση #3: Σελιδοποίηση (paging)



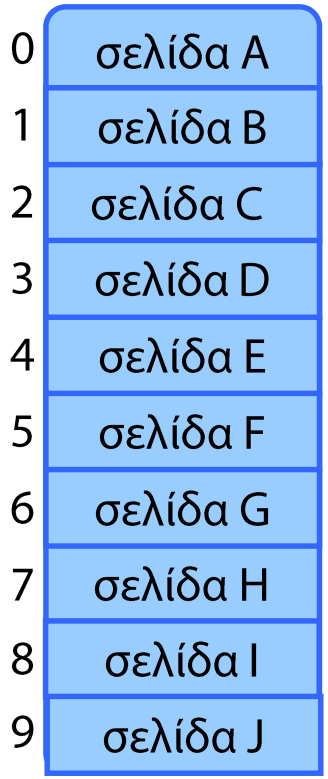
Paging: Λεπτομέρειες υλοποίησης

- Δουλεύουμε με τον εικονικό χώρο διευθύνσεων κάθε διεργασίας
- Χρειάζεται να γνωρίζουμε:
  - Όλο τον **εικονικό χώρο διευθύνσεων** κάθε διεργασίας (κώδικας, δεδομένα, στοίβα, σωρός, αρχεία απεικονισμένα στη μνήμη)
    - **Χάρτης Μνήμης (memory map)** που τηρεί ανά διεργασία το ΛΣ
  - Πώς απεικονίζονται (που βρίσκονται) οι σελίδες του εικονικού χώρου διευθύνσεων στη φυσική μνήμη
    - **Πίνακας σελίδων (page table)** δομή του υλικού (ISA) που γνωρίζει ο επεξεργαστής και τη χειρίζεται σε συνεργασία με το ΛΣ. Χρησιμοποιείται από τον επεξεργαστή σε κάθε εντολή πρόσβασης στη μνήμη για να μεταφράσει εικονικές (λογικές) διευθύνσεις σε φυσικές

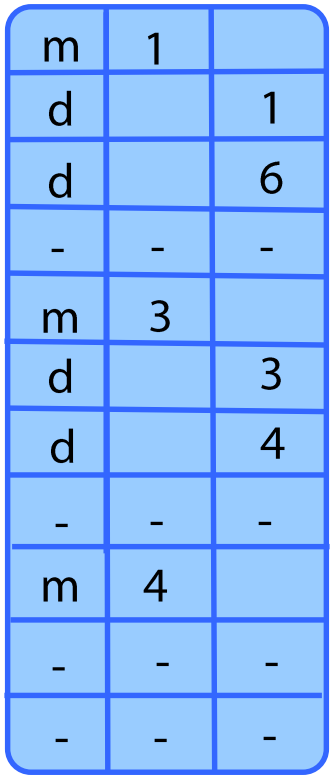
Εικονικός χώρος διευθύνσεων και χάρτης μνήμης

- Ο **μέγιστος εικονικός χώρος** διευθύνσεων είναι πολύ μεγάλος, π.χ. 2^{32}, 2^{48}, 2^{64}, ανάλογα με την αρχιτεκτονική.
- Οι διεργασίες τυπικά χρησιμοποιούν ένα μέρος του μέγιστου εικονικού χώρου που:
  - Δεν είναι απαραίτητα συνεχόμενο
  - Δεν βρίσκεται απαραίτητα ολόκληρο στη μνήμη (μέρη του μπορεί να βρίσκονται στο δίσκο)
  - Κάθε περιοχή του έχει διαφορετικά χαρακτηριστικά (π.χ. δικαιώματα πρόσβασης, κοινή χρήση)
  - Μπορεί να αλλάζει κατά τη διάρκεια εκτέλεσης μιας διεργασίας (malloc, free, function calls, mmap)
- Όλα τα παραπάνω τηρούνται στο χάρτη μνήμης κάθε διεργασίας εντός του ΛΣ

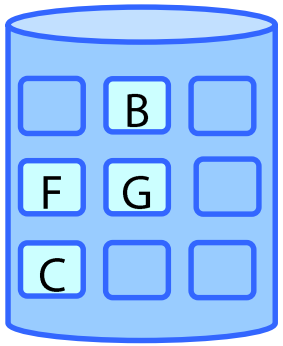
Χάρτης μνήμης



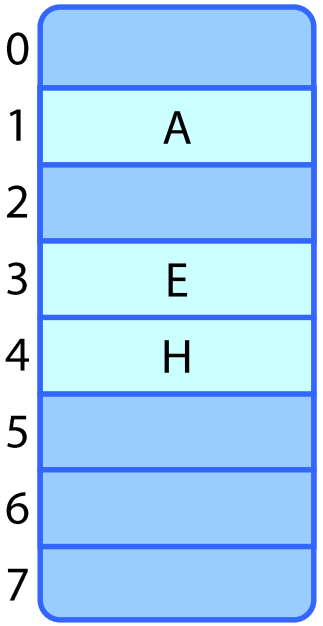
Μέγιστη Εικονική Μνήμη



Χάρτης μνήμης (δομή ΛΣ)



Δίσκος



Φυσική Μνήμη

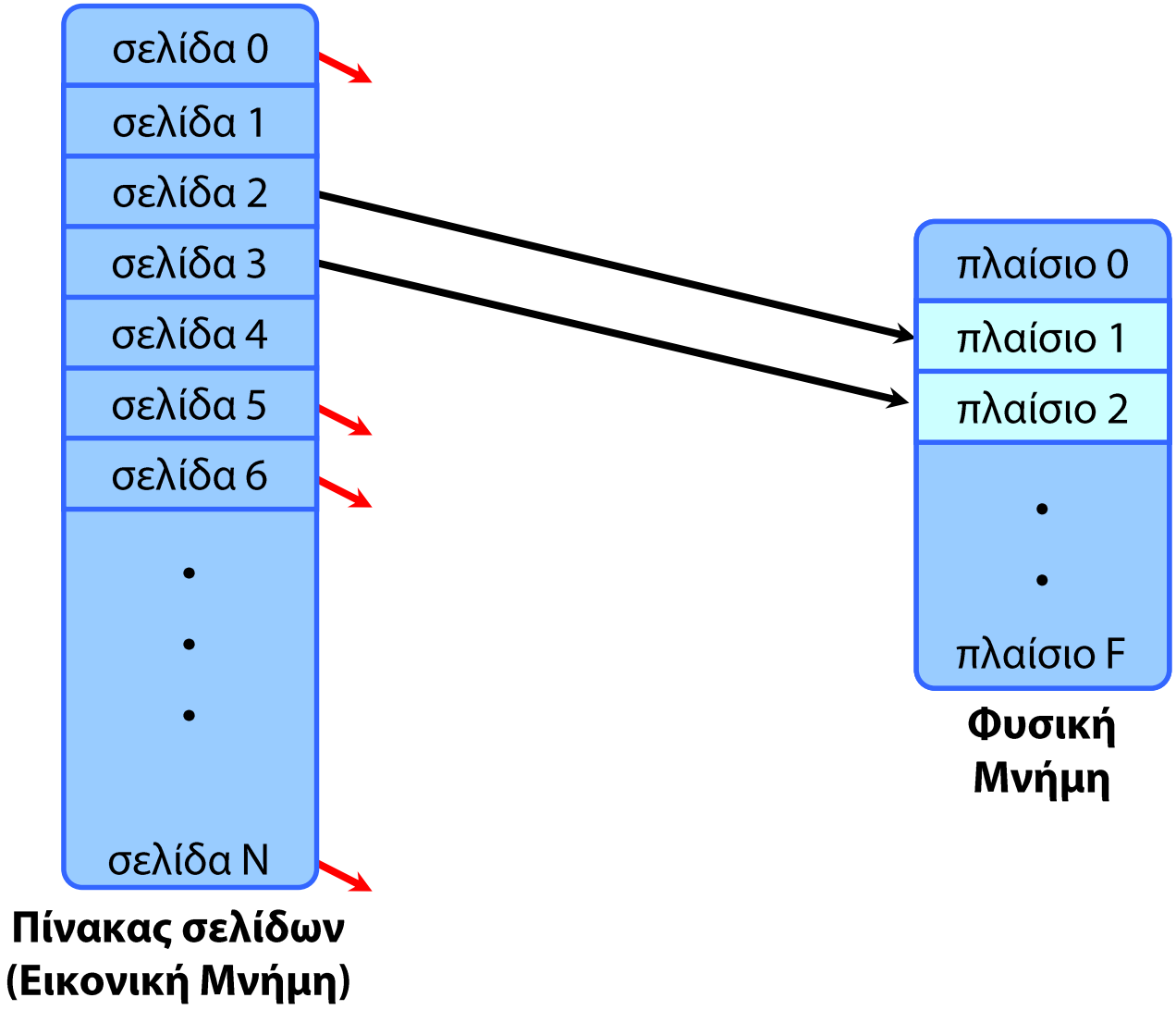
Χάρτης μνήμης

- Ο χάρτης μνήμης δημιουργείται με τη γέννηση της διεργασίας
- Το ΛΣ διαβάζει πληροφορίες από το εκτελέσιμο αρχείο και παραχωρεί κατάλληλο χώρο για τον κώδικα, τα δεδομένα και έναν αρχικό χώρο για τη στοίβα και το σωρό
- Ο χάρτης μνήμης ανανεώνεται αν:
  - Το πρόγραμμα ζητήσει να μεγαλώσει τη στοίβα του (είτε ρητά με system call είτε μέσω function calls, π.χ. με αναδρομικές κλήσεις)
  - Το πρόγραμμα ζητήσει/ελευθερώσει δυναμικά μνήμη στο σωρό (malloc/free)
  - Το πρόγραμμα απεικονίσει ένα αρχείο στη μνήμη
  - Γίνει δυναμική σύνδεση με κάποια εξωτερική βιβλιοθήκη
  - Γίνει αίτημα για κοινή μνήμη
  - Γίνει fork (βλ. COW αργότερα)

Πίνακας σελίδων

- Δομή την οποία συμβουλεύεται ο επεξεργαστής **για τη μετάφραση κάθε πρόσβασης στη μνήμη.**
- Η υλοποίηση του πίνακα σελίδων είναι σύνθετο πρόβλημα.
  - Πρέπει να είναι απλή για να τη χρησιμοποιεί εύκολα το υλικό.
  - Αν είναι πιο σύνθεση, χρειάζεται συνεργασία υλικού / λογισμικού (ΛΣ)
- Τυπική λύση (συστήματα x86):
  - **Απλή λογική:** Για **κάθε διεργασία** τηρείται μία εγγραφή για **κάθε πιθανή σελίδα** του μέγιστου εικονικού χώρου διευθύνσεων (ο επεξεργαστής δεν γνωρίζει το χάρτη μνήμης της διεργασίας και πρέπει να προετοιμαστεί για παν ενδεχόμενο)

Πίνακας σελίδων



Πίνακας σελίδων

- **Όμως:** ο μέγιστος εικονικός χώρος διευθύνσεων είναι μεγάλος
  - Σε **32bit** συστήματα με σελίδες/πλαίσια μεγέθους 4096 bytes χρειαζόμαστε 2^{20} εγγραφές (των 4 bytes η κάθε μία) = **4MB ανά διεργασία**!
    - Για 100 διεργασίες;
    - Για 48bit / 64bit συστήματα;
    - Πού θα βρώ τόσο χώρο για τα page tables;
      - Δεν μπορώ μέσα στον επεξεργαστή (δεν χωράει!)
      - Δεν μπορώ στο δίσκο (είναι απελπιστικά αργός!)
      - Στη μνήμη...

Πίνακες σελίδων στη μνήμη

- Δύο προβλήματα:
  - Οι πίνακες σελίδων καταναλώνουν πολλή μνήμη
  - Η μνήμη είναι αργή σε σχέση με τον επεξεργαστή (για κάθε πρόσβαση στη μνήμη, θα πρέπει να πηγαίνω στη μνήμη για να μάθω τη φυσική διεύθυνση)
- Για να μικρύνω την κατανάλωση μνήμης:
  - **Παρατήρηση:** οι διεργασίες δεν αξιοποιούν συνήθως όλο τον εικονικό χώρο διευθύνσεων
    - πολυεπίπεδα page tables, συνεργασία ΛΣ για την τήρησή τους
      - Όμως τώρα έκανα χειρότερο το πρόβλημα της επίδοσης! 3-5 προσβάσεις στη μνήμη για να κάνω μία μετάφραση!!!
- Για να βελτιώσω την επιδοση:
  - Caching to the rescue!
    - Ειδικές κρυφές μνήμες (Translation Lookaside Buffers) εντός του επεξεργαστή για να κρατούν συχνά χρησιμοποιούμενες μεταφράσεις

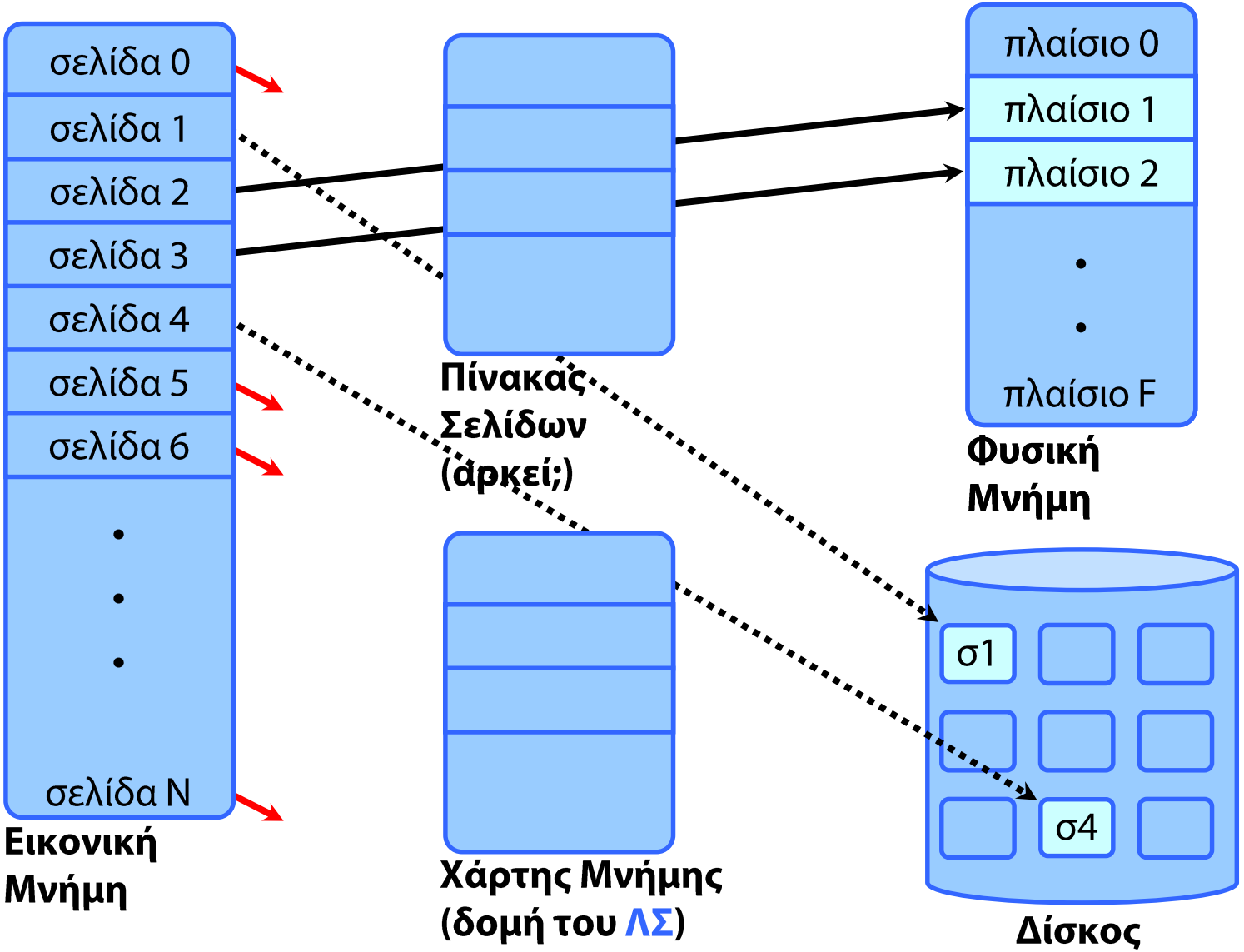
Πίνακες σελίδων στη μνήμη (σύνοψη)

- Οι πίνακες σελίδων είναι σύνθετο και κρίσιμο πρόβλημα γιατί χρειάζονται σε κάθε εντολή ενός προγράμματος
- Απλοποιώντας, μπορούμε να θεωρήσουμε ότι:
  - Με κατάλληλη χρήση κρυφών μνημών (TLBs) η συντριπτική πλειοψηφία των μεταφράσεων μπορεί να γινει εντός του επεξεργαστή σε λίγους κύκλους (make the common case fast)
  - Αν δεν μπορεί να εξυπηρετηθεί από τον TLB ο επεξεργαστής «αυτόματα» διασχίζει το πολυεπίπεδο page table στη μνήμη (page table walk) που μπορει να έχει κόστος εκατοντάδες κύκλους

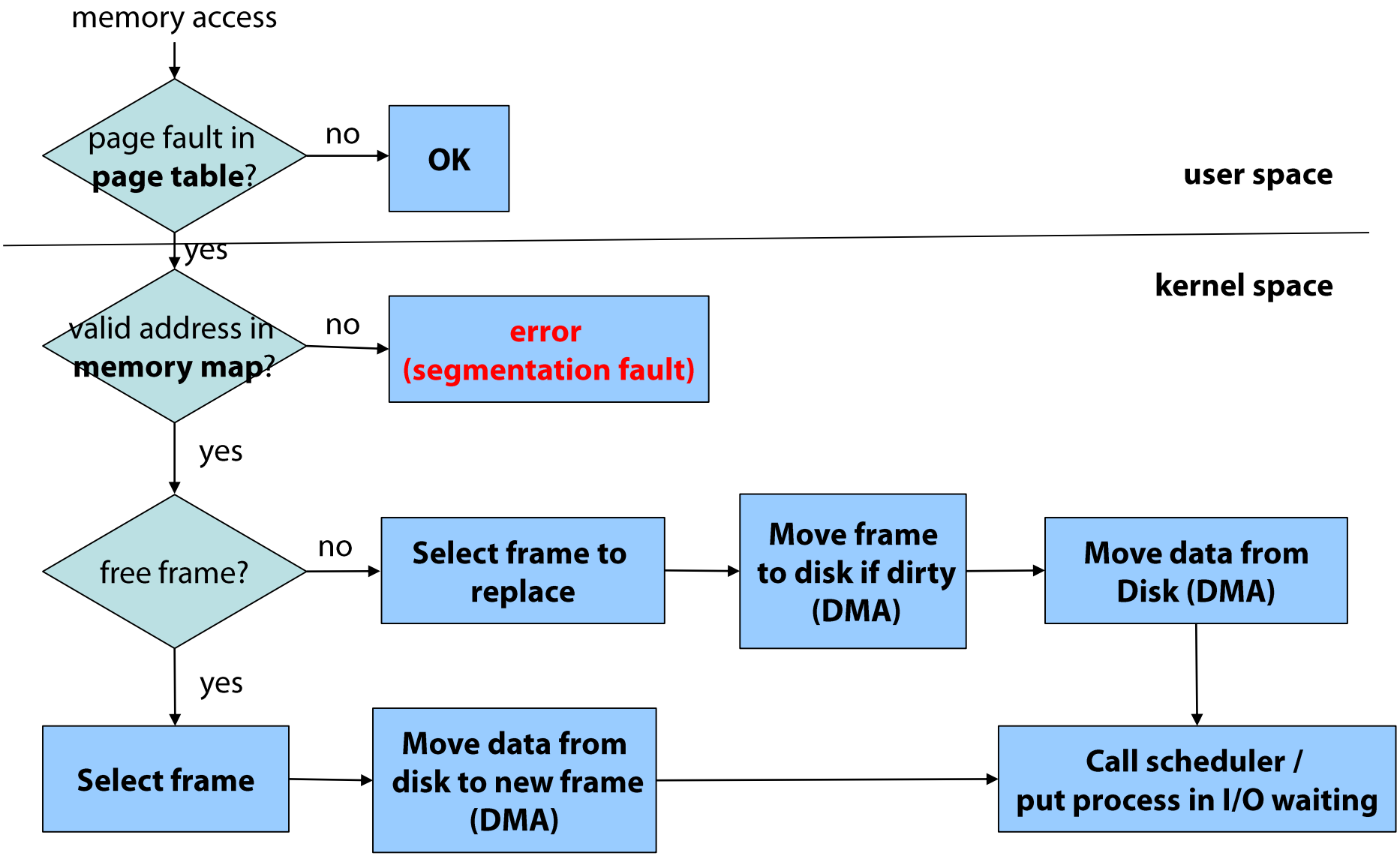
Εικονική Μνήμη με Σελιδοποίηση

- Μία διεργασία θα λάβει πλαίσιο φυσικής μνήμης όταν το χρειαστεί: **Demand paging**
- Κάθε πρόσβαση στη φυσική μνήμη περνάει από τον πίνακα σελίδων.
  - Αν υπάρχει έγκυρη μετάφραση -> OK
  - Αν δεν υπάρχει έγκυρη μετάφραση έχουμε **page fault,** ο έλεγχος μεταφέρεται στο ΛΣ και διακρίνουμε δύο περιπτώσεις με τη βοήθεια του Χάρτη Μνήμης
    - Η διεύθυνση είναι έγκυρη αλλά όχι στην Κύρια Μνήμη
    - Η διεύθυνση δεν είναι έγκυρη

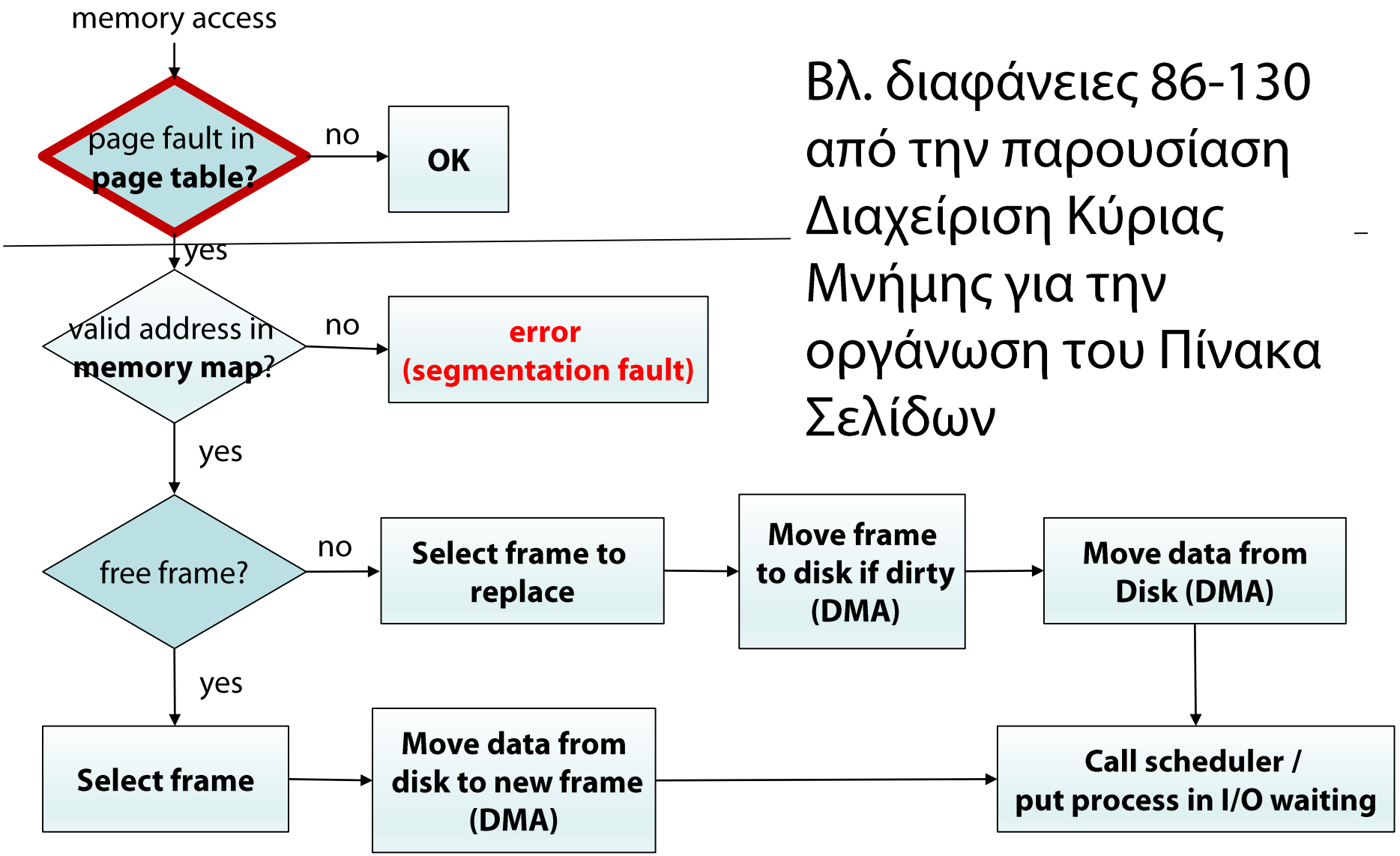
Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα



Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα

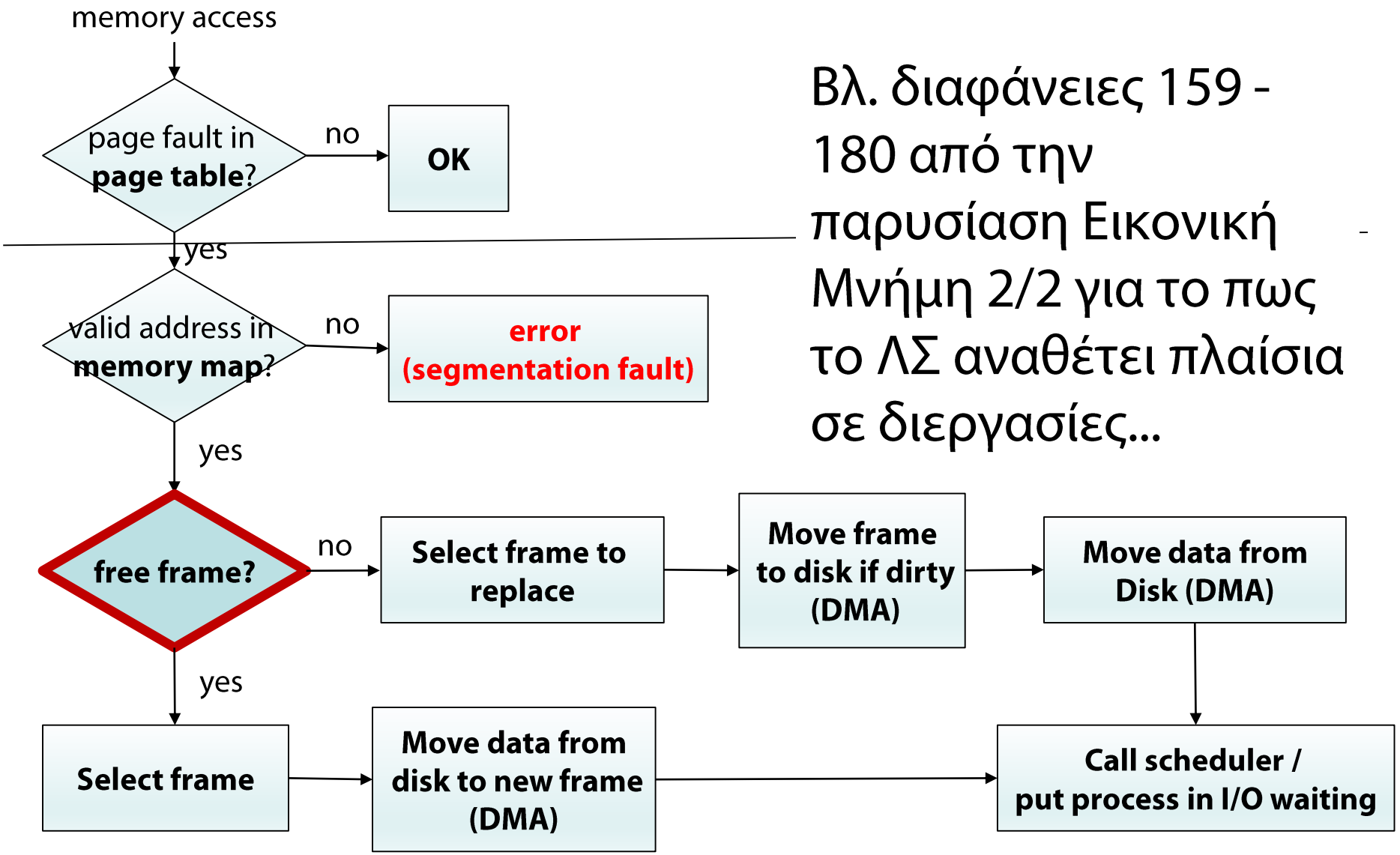


Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα



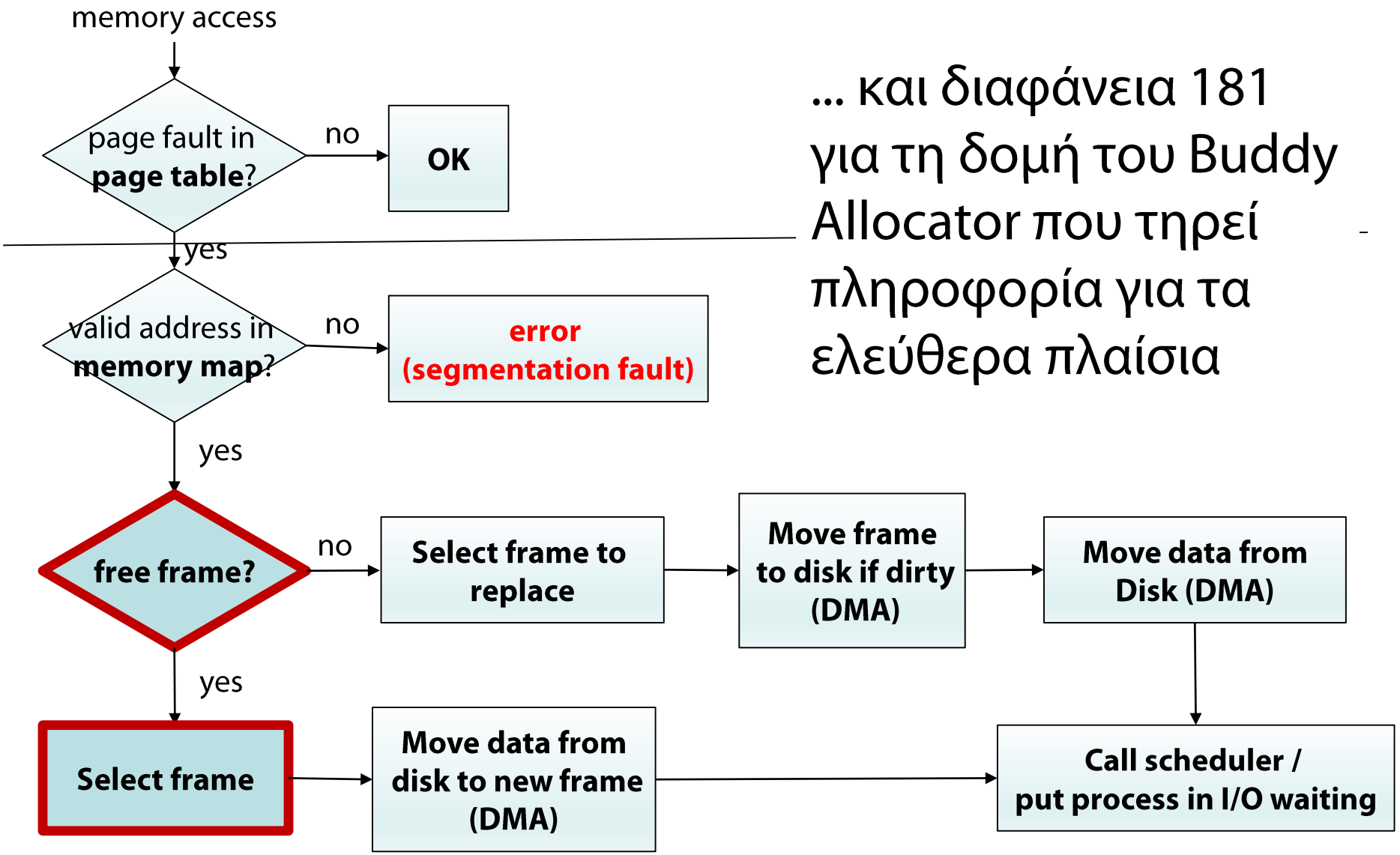
Βλ. διαφάνειες 86-130 από την παρουσίαση Διαχείριση Κύριας Μνήμης για την οργάνωση του Πίνακα Σελίδων

Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα



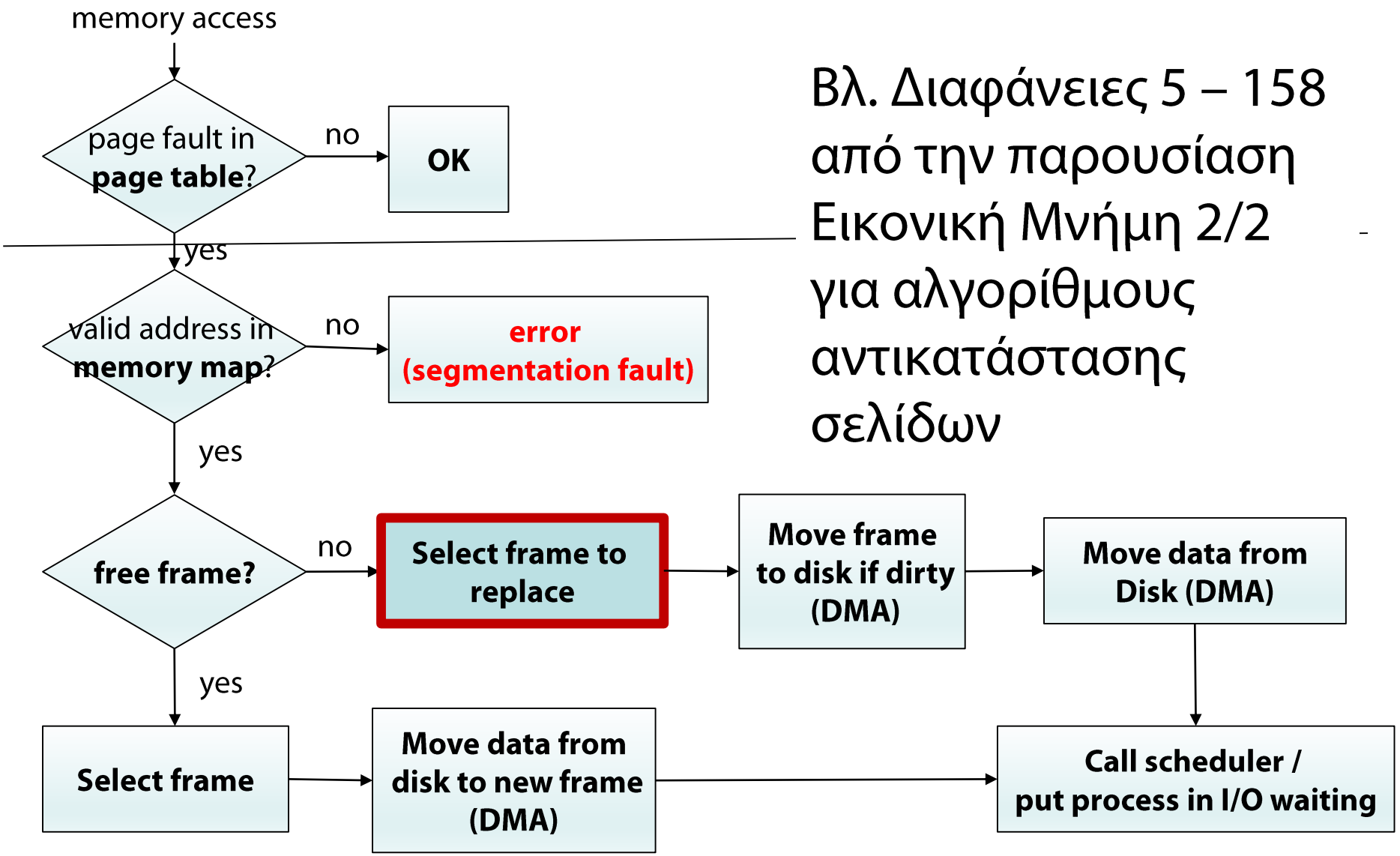
Βλ. διαφάνειες 159 - 180 από την παρουσίαση Εικονική Μνήμη 2/2 για το πως το ΛΣ αναθέτει πλαίσια σε διεργασίες...

Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα



... και διαφάνεια 181 για τη δομή του Buddy Allocator που τηρεί πληροφορία για τα ελεύθερα πλαίσια

Εικονική Μνήμη με Σελιδοποίηση: Η συνολική εικόνα



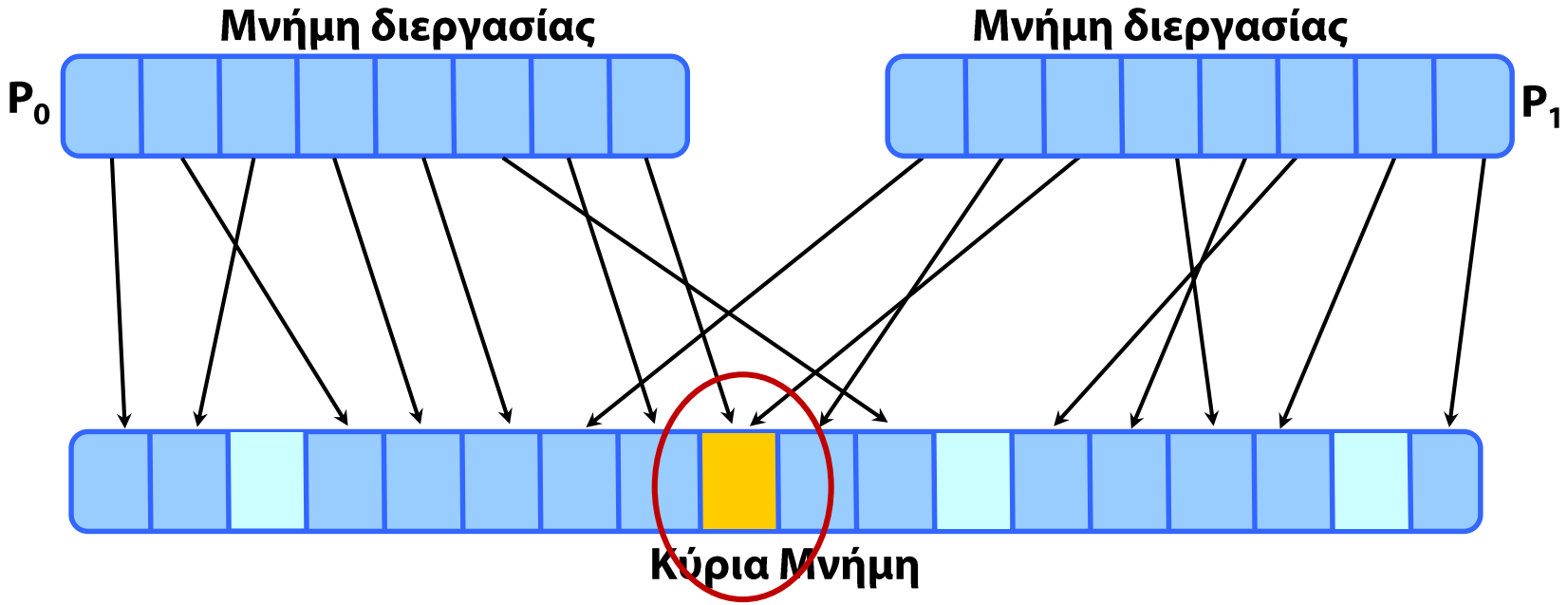
Βλ. Διαφάνειες 5 – 158 από την παρουσίαση Εικονική Μνήμη 2/2 για αλγορίθμους αντικατάστασης σελίδων

Επιπλέον λειτουργικότητα της σελιδοποίησης

- Λεπτομερή (ανά σελίδα) δικαιώματα πρόσβασης (βλ. διαφάνειες 102 – 107 παρουσίασης διαχείριση ΚΜ). Προστασία για:
  - Να μην εγγραφούν σταθερές
  - Να μην εκτελεστεί κώδικας στην περιοχή των δεδομένων
  - Null pointer dereference (το NULL απεικονίζεται σε ειδικές σελίδες χωρίς δικαίωμα πρόσβασης)

Επιπλέον λειτουργικότητα της σελιδοποίησης

- **Κοινά** δεδομένα και βιβλιοθήκες χρόνου εκτέλεσης (π.χ. libc) αλλά και ίδιο το ΛΣ, βλ. διαφάνειες 42-49 παρουσίαση Εικονική Μνήμη 1/2.



Επιπλέον λειτουργικότητα της σελιδοποίησης

- Copy-On-Write (COW)
  - **Θυμηθείτε:** στη fork η (εικονική) μνήμη του παιδιού είναι αντίγραφο της μνήμης του πατέρα.
  - Χρειαζόμαστε πραγματικά να φτιάξουμε ένα ολόκληρο αντίγραφο; Η πρόσβαση στη μνήμη κοστίζει!
    - Όχι για κώδικα, σταθερές και read-only δεδομένα
    - Όταν μετά τη fork ακολουθεί execv! (η πιο συχνή περίπτωση)
  - Θα αντιγραφούν μόνο οι σελίδες στις οποίες μία διεργασία επιχειρήσει να γράψει (και θα γίνει ακριβώς κατά την απόπειρα εγγραφής)
  - Μετά τη fork δεν γίνονται αντιγραφές, ενημερώνονται ο χάρτης μνήμης και ο πίνακας σελίδων (π.χ. read-only και reference counting)

Επιπλέον λειτουργικότητα της σελιδοποίησης

- Απεικόνιση αρχείων στη μνήμη
  - System call **mmap**
  - Απεικονίζει ένα αρχείο στην εικονική μνήμη της διεργασίας και επιστρέφει ένα δείκτη στη μνήμη (π.χ. στην αρχή του αρχείου ή και αλλού)
  - Πρόσβαση στο αρχείο με εντολές ανάγνωσης/εγγραφής στη μνήμη

Εικονική Μνήμη με σελιδοποίηση

Τέλική αξιολόγηση της λύσης:

- Καλή χρήση της μνήμης
  - Μπορεί να αξιοποιηθεί κάθε πλαίσιο της μνήμης
  - Μεγάλες διεργασίες μπορούν να εκτελεστούν
  - Δυναμική παραχώρηση και επιστροφή μνήμης
- Υψηλή επίδοση
  - Μικρός χρόνος μετάφρασης με τη βοήθεια του TLB
    - Όμως η επιβάρυνση μπορεί να είναι σημαντική για TLB misses
  - Ελαχιστοποίηση χρήσης του δίσκου
- Απλότητα
  - Χρειάζεται σοβαρή υποστήριξη από το υλικό
  - Πολύπλοκο σύστημα που περιλαμβάνει συνεργασία υλικού / λογισμικού
  - **Όμως:** Τα παραπάνω δεν γίνονται ορατά στον τελικό χρήστη (προγραμματιστή)

Ερωτήσεις

Τι συμβαίνει στο σύστημα Εικονικής Μνήμης και γενικά στο ΛΣ όταν το πρόγραμμά μου εκτελεί:

- κλήση συνάρτησης ή υπορουτίνας;
- malloc / new
- πρόσβαση σε δεδομένα στο σωρό
- mmap
- fork

Ερωτήσεις;

και στη λίστα:

OS@lists.cslab.ece.ntua.gr