# Λειτουργικά Συστήματα (HY321)

## Διάλεξη 8:
## Σελιδοποίηση & Swapping

**Χρήστος Δ. Αντωνόπουλος**
**6/4/2017**

**Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας**

# Από τα Προηγούμενα…

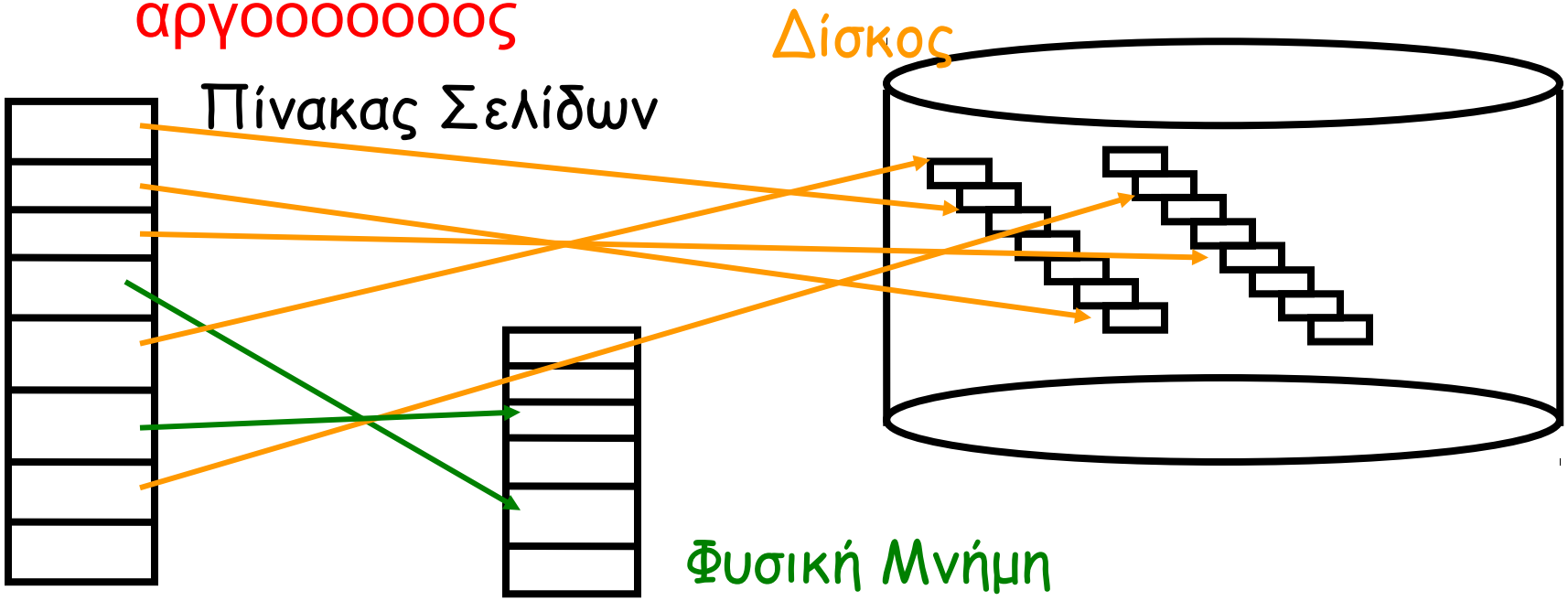
- Φυσική μνήμη
  - Έλλειψη προστασίας
  - Περιορισμένο μέγεθος
  - Συνεχείς ή κατά κανόνα συνεχείς περιοχές
  - Διαμοίραση ορατή στα προγράμματα
    - Αλλά και εύκολη
- Ιδεατή μνήμη
  - Απομόνωση διαφορετικών προγραμμάτων
  - Αδιαφανής: Δεν ξέρουμε που βρισκόμαστε στη φυσική μνήμη
  - Μη συνεχείς περιοχές στη φυσική μνήμη
  - Είναι δυνατόν να διαμοιρασθούν κώδικας & δεδομένα

# Swapping

- Η «αγαθή» αντιμετώπιση
  - Φόρτωσε όλη τη διεργασία στη φυσική μνήμη
  - Εκτέλεσέ τη
  - Τερμάτισε
- Προβλήματα;
  - Αργό (αν η διεργασία είναι «μεγάλη»)
  - Κακή χρήση της μνήμης (αν η διεργασία δε χρησιμοποιεί στην πραγματικότητα όλο το χώρο διευθύνσεών της)
- Λύσεις:
  - Swapping
    - Κράτησε στην κύρια μνήμη μόνο τις σελίδες που χρησιμοποιούνται
  - Swapping με βάση τη ζήτηση
    - Φόρτωσε στη φυσική μνήμη μόνο τις σελίδες που ζητάει η διεργασία (on demand)
- Μηχανισμός:
  - Αξιοποίησε τους μηχανισμούς ιδεατής μνήμης, ώστε κάποιες σελίδες ιδεατής μνήμης να είναι αποθηκευμένες στην κύρια μνήμη και κάποιες στο δίσκο

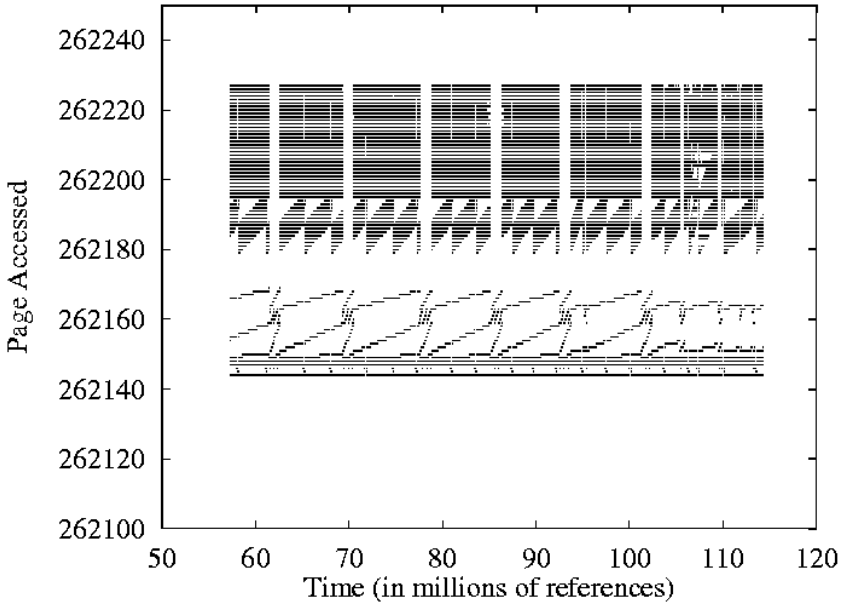
# Swapping (Όταν Κοιτάζω από Ψηλά…)

- Οι ιδεατές σελίδες αντιστοιχούν σε:
  - Μνήμη (0.00559€ / MB), γρήγορη, μικρή
  - Δίσκος (0.00003795€ / MB), μεγάλος, αργόοοοοοος



# Swapping: Η Μεγάλη Απάτη

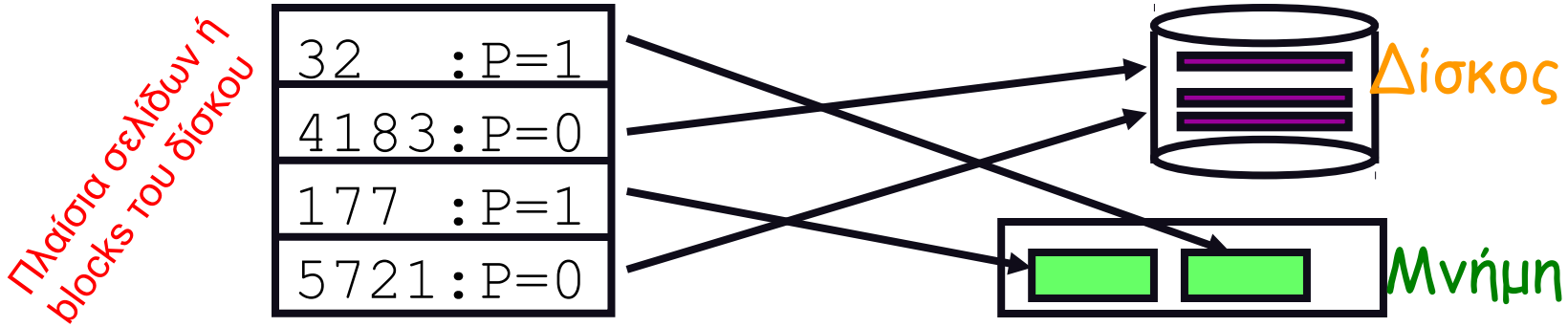
- Μπορώ να έχω:
  - Ιδεατή μνήμη μεγάλη σαν δίσκο και γρήγορη σαν κύρια μνήμη;
    - Ο κανόνας 80/20 και πάλι…
    - Κράτησε το 20% στη μνήμη και το υπόλοιπο 80% στο δίσκο
    - Και ποιο είναι το 20%;
      - Κοίταξε τη συμπεριφορά στο πρόσφατο παρελθόν
  - Ενάντια στη διαίσθηση:
    - 1 ακόμα επίπεδο ανακατεύθυνσης, και όμως η επίδοση βελτιώνεται
    - Τυπική ευστοχία στην κύρια μνήμη: 99.9%




Μοτίβο προσπέλασης στη μνήμη του Gzip

# Ο Μηχανισμός του Swapping (και πάλι…)

- Επέκταση κάθε γραμμής του πίνακα σελίδων με ένα επιπλέον bit (“παρούσα”)
  - Αν η σελίδα στη μνήμη παρούσα = 1,  αν στο δίσκο παρούσα = 0
  - Όταν παρούσα = 1 οι «μεταφράσεις» δουλεύουν όπως ξέρουμε
  - Όταν παρούσα = 0, η «μετάφραση» προκαλεί σφάλμα σελίδας (page fault).




- Τι συμβαίνει στα σφάλματα σελίδων;
  - Το ΛΣ βρίσκει ένα άδειο πλαίσιο στη φυσική μνήμη ή αδειάζει ένα (ποιο;;;)
  - Κάνει μια αίτηση στο δίσκο για να διαβάσει τα περιεχόμενα της ιδεατής σελίδας από το δίσκο και να τα τοποθετήσει στο άδειο πλαίσιο
  - Μπλοκάρει τη διεργασία και εκτελεί μια άλλη
  - Όταν τελειώσει η ανάγνωση από το δίσκο: παρούσα = 1, η διεργασία πίσω στην ουρά των έτοιμων προς εκτέλεση

# Ο Μηχανισμός του Swapping (Λίγο Ακόμα…)

- Και τι θα κάνω με τις σελίδες που πρέπει να διώξω (αντικαταστήσω);
  - Έχουν αλλάξει;
    - Ε, τότε πρέπει να γραφτούν στο δίσκο
  - Δεν έχουν αλλάξει;
    - Πέταξέ τες, υπάρχει προηγούμενο αντίγραφο στο δίσκο
  - Και που να ξέρω;
    - Η αγαπημένη μας λύση: Το λογαριασμό στον κατασκευαστή υλικού
      - Ένα ακόμα bit σε κάθε γραμμή του πίνακα σελίδων (modified)
        - Όταν η σελίδα γραφτεί γίνεται 1 (από το υλικό)
        - Όσο η σελίδα δεν έχει μεταβληθεί είναι 0

| | | |
|---|---|---|
| 32 | P=1 | M=1 |
| 4183 | P=0 | M=0 |
| 177 | P=1 | M=0 |
| 5721 | P=0 | M=0 |

# Προβλήματα, Προβλήματα, Προβλήματα…

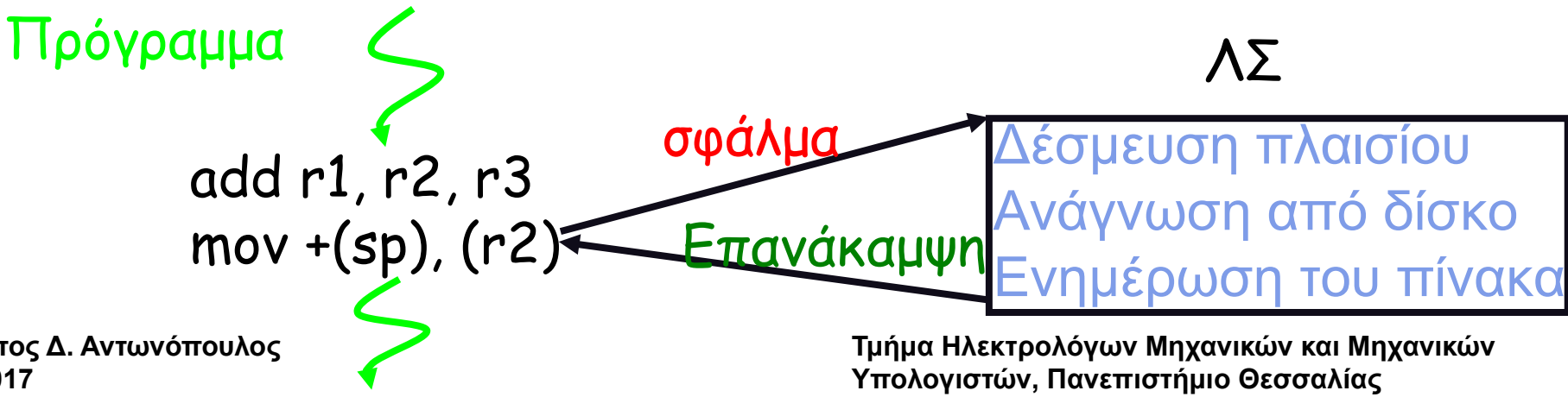
- Πώς να επανεκκινήσω μια διεργασία μετά από σφάλμα σελίδας;
  - Απλό: Πρέπει να σώσω την κατάσταση και να την αποκαταστήσω μετά την εξυπηρέτηση του σφάλματος.
  - Και αν… η διεργασία ήταν στη μέση της εκτέλεσης μιας εντολής;

- Τι να φέρω από το δίσκο;
  - Μόνο τη σελίδα που ζητήθηκε, ή και καμία ακόμα;

- Τι να διώξω από την κύρια μνήμη;
  - Όπως όλες οι cache, έτσι και η κύρια μνήμη είναι πολύ μικρή. Τι να διώξω κάθε φορά που θέλω να φέρω μια σελίδα;
  - Θέλω να ξέρω τη μελλοντική συμπεριφορά !!!

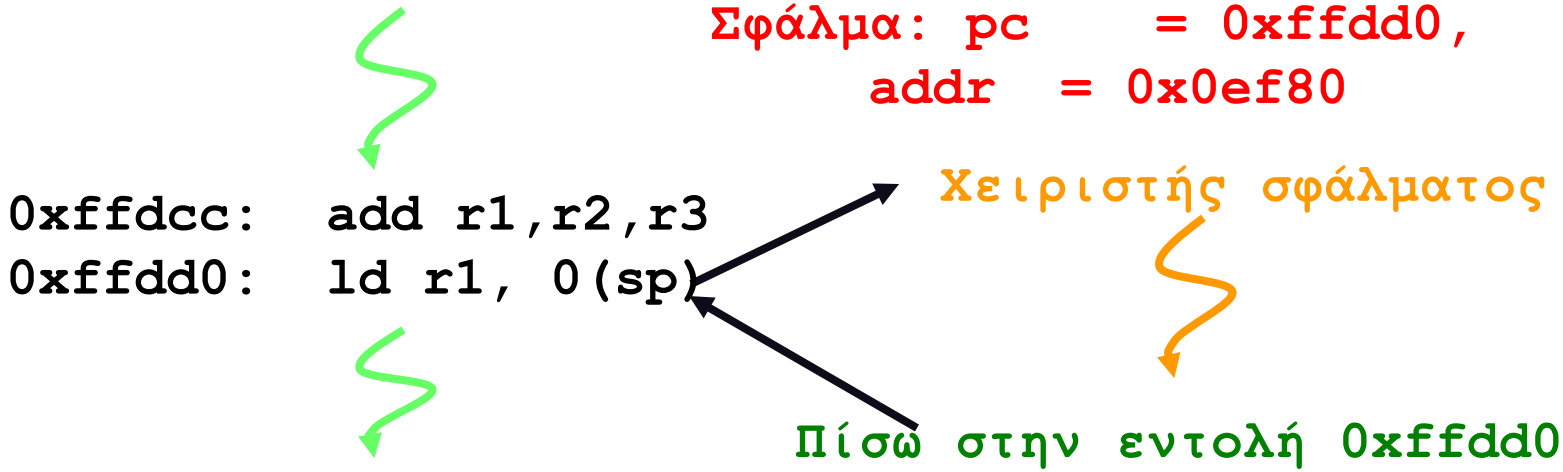
# Επανάκαμψη μετά από Σφάλμα Σελίδας

- Μπορεί το σφάλμα να συνέβη στη μέση της εκτέλεσης μιας εντολής!
  - Ο χρήστης δεν πρέπει να καταλάβει τίποτα για το σφάλμα σελίδας
  - Μπορούμε να παραλείψουμε την εντολή που προκάλεσε το σφάλμα;
    - Μπα… μάλλον όχι… Κάποια δουλειά κάνει στο πρόγραμμα
  - Μπορούμε να την ξαναεκτελέσουμε εξ' αρχής;
    - Όχι, αν έχει αλλάξει μερικώς την κατάσταση της μηχανής.
  - Μπορούμε να αποφασίσουμε κοιτάζοντας τι κάνει η εντολή;
    - Ίσως να μην είναι σαφές πότε ακριβώς συνέβη το σφάλμα



# Την Ευθύνη στο Υλικό (και πάλι…)

- Συστήματα RISC: Απλά τα πράγματα
  - Συνήθως οι εντολές δεν αλλάζουν την κατάσταση της μηχανής πριν γίνουν όλες οι αναφορές
  - Αρκεί να ξέρω την εντολή που προκάλεσε το σφάλμα σελίδας και τη διεύθυνση στην οποία αντιστοιχεί το σφάλμα




- Συστήματα CICS: Πρόβλημα…
  - Πιο πολύπλοκες εντολές
  - Πολλαπλές αναφορές στη μνήμη
  - Πιθανόν τη στιγμή του σφάλματος η κατάσταση της μηχανής να έχει αλλάξει μερικώς

# Τι να Φέρω; (και Πότε;)

- Ποιες σελίδες να φέρω στη μνήμη και πότε;
  - Το πρόβλημα που έχουμε με όλες τις caches: χρειαζόμαστε γυάλινη σφαίρα (γνώση του μέλλοντος)
- Ας πει ο χρήστης
  - Αξιόπιστο;
  - Μερικά ΛΣ υποστηρίζουν προ-μεταφορές
- Εύκολο κόλπο κατά το χρόνο εκτέλεσης: σελιδοποίηση που ακολουθεί τη ζήτηση
  - Φόρτωσε 1/περισσότερες αρχικές σελίδες. Εκτέλεσε. Φόρτωσε τις άλλες σελίδες όταν χρειαστούν (σφάλμα σελίδας).
    - Συνέπειες στο χρόνο εκτέλεσης;
    - Συνέπειες στην αξιοποίηση της μνήμης;
  - Τα περισσότερα συστήματα το χρησιμοποιούν

ld αρχικές σελ → ld σελ → ld σελ → ld σελ → ...

- Κι άλλο κόλπο: προ-μεταφορά. Φέρε τη σελίδα και τη γειτονιά της
  - Γιατί;

# Ποια να Διώξω;

- Τυχαία: Διάλεξε μια σελίδα
  - + Αποφεύγουμε τη χειρότερη περίπτωση, απλό
  - - Αποφεύγουμε και την καλύτερη περίπτωση ☹
- FIFO: Διώξε την παλαιότερη σελίδα
  - + Δίκαιο. Όλες οι σελίδες μένουν στη μνήμη το ίδιο διάστημα
  - - Αγνοεί το μοτίβο χρήσης των σελίδων
- MIN (βέλτιστο):
  - Διώξε τη σελίδα που δεν πρόκειται να χρησιμοποιηθεί για το μεγαλύτερο διάστημα.
  - Και που να ξέρω;
    - Μη πρακτικό, αλλά πάντως καλή ιδέα
- Τη λιγότερο πρόσφατα χρησιμοποιηθείσα (LRU).
  - Διώξε τη σελίδα που έχει να χρησιμοποιηθεί το μεγαλύτερο διάστημα.
  - Αν παρελθόν = μέλλον;
    - LRU = MIN.