Λειτουργικά Συστήματα (ΗΥ321)

Διάλεξη 7:

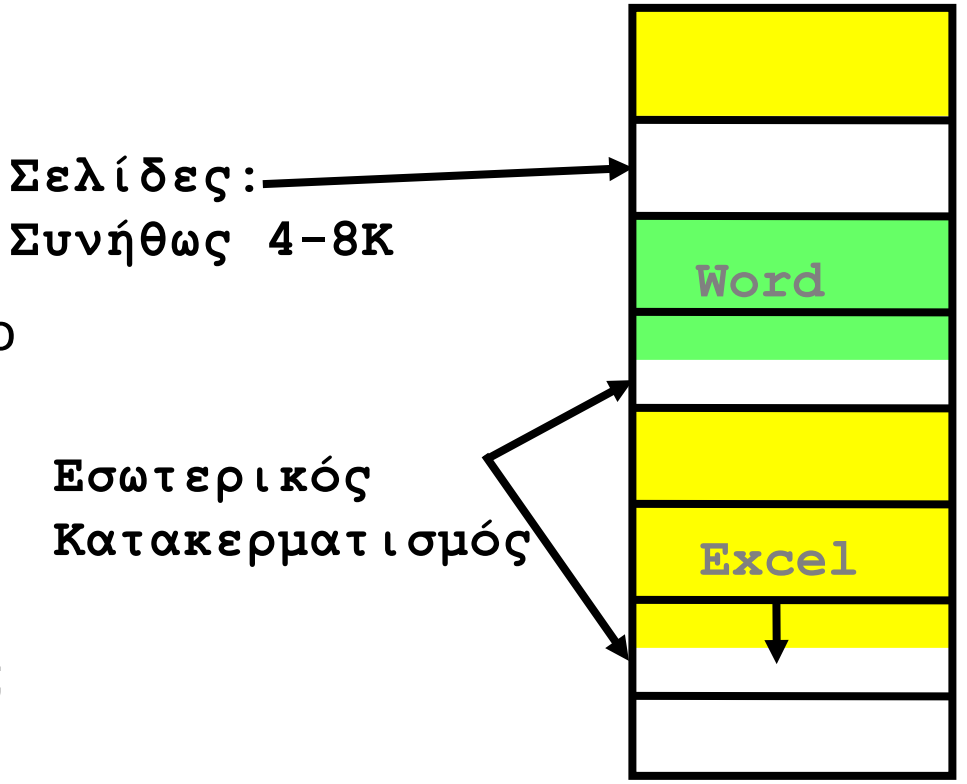
Εικονική Μνήμη – Σελιδοποίηση & Πίνακες Σελίδων

Χρήστος Δ. Αντωνόπουλος
30/3/2017

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

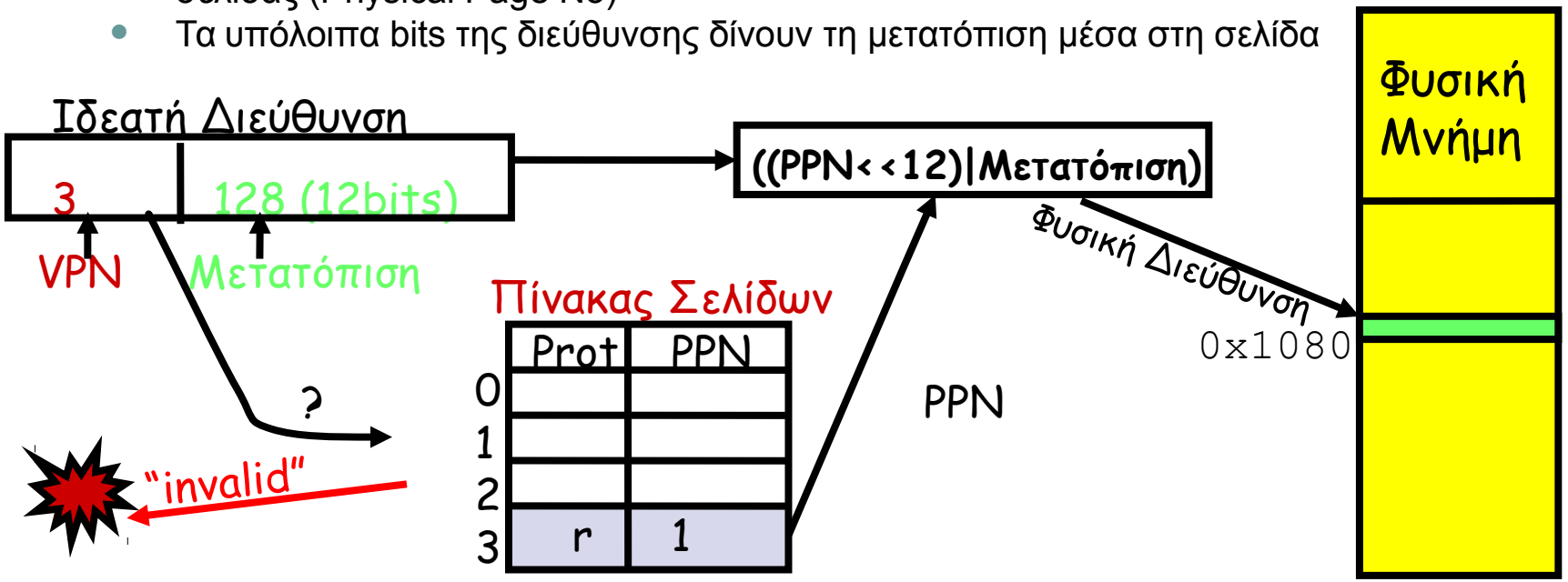
Ιδεατή Μνήμη Βασισμένη σε Σελίδες (Σελιδοποίηση)

- Σπάσε τη μνήμη σε κομματάκια σταθερού μεγέθους (σελίδες)
- Δίλλημα:
  - + Εξαφανίζεται ο εξωτερικός κατακερματισμός
  - + Δεν είναι ανάγκη να έχουμε συνεχή περιοχή στη μνήμη για το τμήμα
  - + Απλούστερη δέσμευση/αποδέσμευση/μεταφορά μνήμης στο δίσκο
  - - Εσωτερικός κατακερματισμός
    - Πόσος κατά μέσο όρο ανά τμήμα;
  - - Ένα ακόμα επίπεδο ανακατεύθυνσης



Μηχανισμοί Σελιδοποίησης

- Η μνήμη χωρίζεται σε κομμάτια ίδιου μεγέθους (σελίδες)
- Κάθε διεργασία έχει πίνακα σελίδων (αποθηκεύεται και αυτός στη μνήμη της) που αντιστοιχεί αριθμούς ιδεατών σελίδων σε αριθμούς φυσικών σελίδων
  - Ο πίνακας περιέχει επίσης και bits δικαιωμάτων (Prot)
    - R / W / X / Valid
- Διαδικασία μετάφρασης:
  - Κάποια bits στη διεύθυνση υποδεικνύουν τον αριθμό ιδεατής σελίδας (Virtual Page No)
  - Αυτός χρησιμοποιείται σαν δείκτης στον πίνακα, για να βρεθεί ο αριθμός πραγματικής σελίδας (Physical Page No)
  - Τα υπόλοιπα bits της διεύθυνσης δίνουν τη μετατόπιση μέσα στη σελίδα



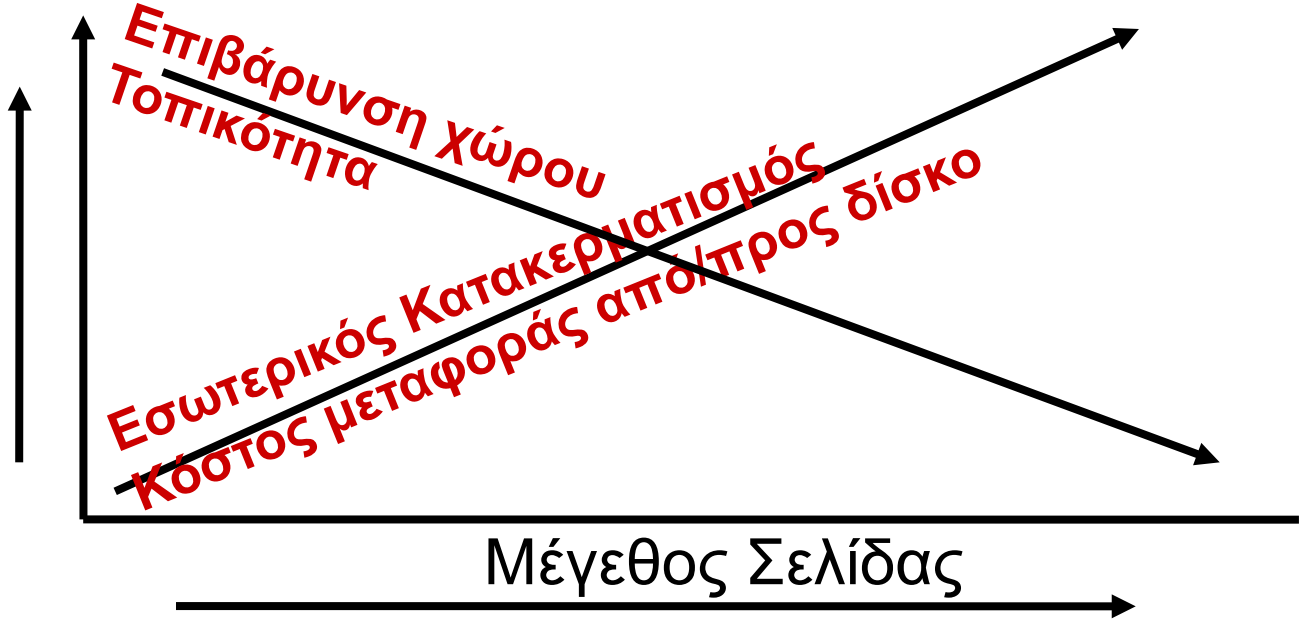
Παράδειγμα

- Intel IA32 (32 bits address space)
  - 12 bits για μετατόπιση
    - Πόσο μεγάλες σελίδες;
  - 20 bits για VPN (αριθμό ιδεατής σελίδας)
    - Πόσες ιδεατές σελίδες;

Σελιδοποίηση: Πανάκεια;

- Πλεονεκτήματα σελίδων
  - Εύκολη δέσμευση
    - Λίστα από ελεύθερες σελίδες
    - Φέρε την πρώτη
  - Εύκολη μεταφορά στο δίσκο
    - Όλες οι σελίδες ίδιου μεγέθους
    - Συχνά όσο ένα block στο δίσκο
- Μειονεκτήματα σελίδων
  - Περισσότερος χώρος για διαχείριση
    - Πόσα δεδομένα για τη διαχείριση ενός τμήματος από 0x0000 έως 0xffff;
    - Πόσα δεδομένα για τη διαχείριση της ίδιας περιοχής με σελιδοποίηση και σελίδες των 4K;
  - Άλλα μειονεκτήματα (… για όταν μεγαλώσετε)
    - Απεικόνιση στις μνήμες cache / συγκρούσεις

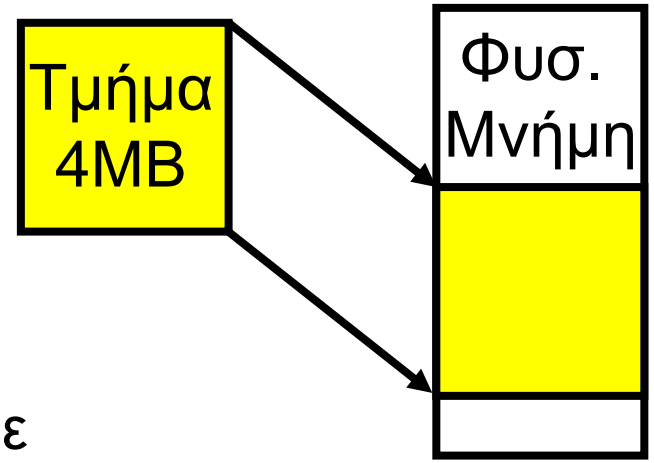
Και … πόσο μεγάλες σελίδες;



- Εσωτερικός κατακερματισμός σε μεγάλα μεγέθη σελίδας:
  - Δεν είναι συνήθως πρόβλημα
  - Εκτός από τη μνήμη των πινάκων σελίδων…

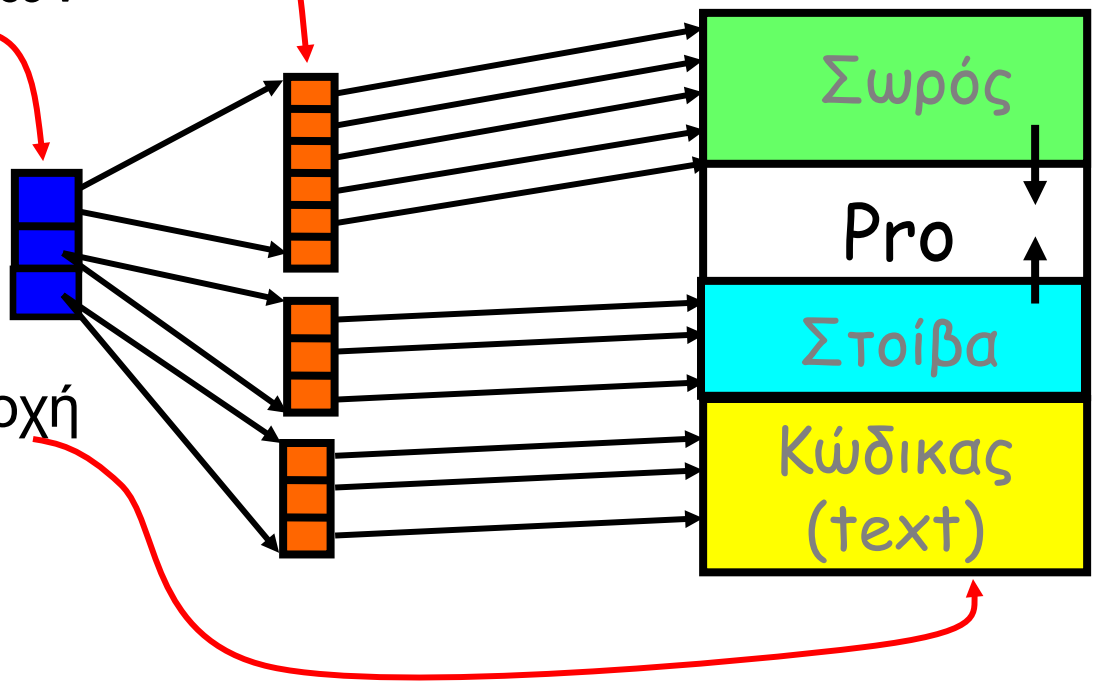
Σελιδοποίηση – Τμηματοποίηση: Ας τα βάψουμε μαύρα

- Προβλήματα… προβλήματα…
  - Σελιδοποίηση
    - Απλός πίνακας αντιστοίχισης
    - Μεγάλη επιβάρυνση σε χώρο
  - Τμηματοποίηση
    - Απαίτηση για συνεχείς περιοχές φυσικής μνήμης / δίσκου για κάθε τμήμα
      - Προβληματική η σελιδοποίηση: Ή όλο το τμήμα στη μνήμη ή όλο εκτός



Σελιδοποίηση + Τμηματοποίηση;;;

- Μνήμη προγράμματος σελιδοποιημένη (χρήση πίνακα σελίδων)
- Μνήμη πίνακα σελίδων τμηματοποιημένη (χρήση πίνακα τμημάτων)
- Και μια αλλαγή ορολογίας:
  - Τμήμα: Συνεχής περιοχή ιδεατών διευθύνσεων
    - Τμήμα περιέχει >=1 σελίδες



Σελιδοποίηση & Τμηματοποίηση

- Η σελιδοποιημένη ιδεατή μνήμη επιτρέπει τμήματα από μη συνεχείς περιοχές φυσικής μνήμης
  - Ευέλικτη διαδικασία διαχείρισης μνήμης
  - Μέρος του τμήματος μπορεί να είναι στο δίσκο
- Τμηματοποίηση: Εύκολη (& φθηνή) διαχείριση της μνήμης
  - Πίνακες σελίδων αντιστοιχούν σε «λογικά» τμήματα της μνήμης (π.χ. κώδικας, σωρός, στοίβα)
    - Ενίοτε αυτά τα τμήματα είναι μεγάλα (άρα και οι πίνακες σελίδων)
    - Αν γίνουν οι πίνακες σελίδων πολύ μεγάλοι;
- Από σελιδοποίηση σε σελιδοποίηση + τμηματοποίηση : ισοδύναμο του από 1 τμήμα σε πολλαπλά
  - Αντί για έναν πίνακα σελίδων πολλαπλοί, ο καθένας συνοδευόμενος από βάση (του πίνακα) και όριο (του πίνακα)

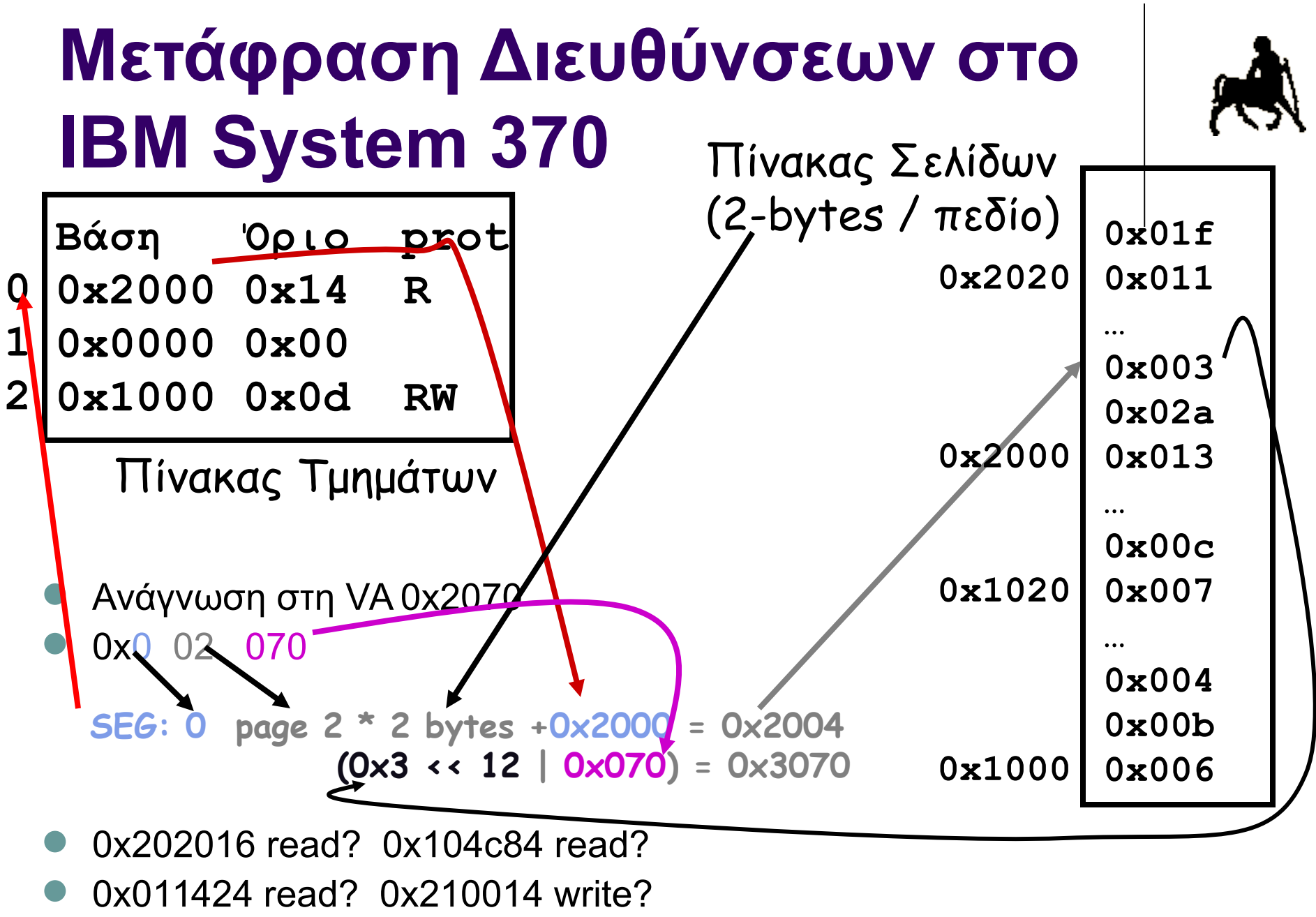
Ένα Παράδειγμα από το Παρελθόν: IBM System 370

- Χώρος ιδεατών διευθύνσεων 24 bits (αρχισαμεεεεε)

# Τμήμ. (4 bits)	# Σελίδας (8 bits)	Μετατόπιση στη σελίδα (12 bits)

- Οι απεικονίσεις:
  - Πίνακας τμημάτων : Αριθμό τμήματος σε φυσική διεύθυνση της βάσης και όριο του πίνακα σελίδων για το αντίστοιχο τμήμα
  - Πίνακας σελίδων : Αριθμό ιδεατής σελίδας σε αριθμό φυσικής σελίδας (12 bits)
    - Ενώνουμε (κολλάμε) τον αριθμό φυσικής σελίδας με τη μετατόπιση και έχουμε την τελική φυσική διεύθυνση

Μετάφραση Διευθύνσεων στο IBM System 370



- 0x202016 read? 0x104c84 read?
- 0x011424 read? 0x210014 write?

Λίγες Λεπτομέρειες για Σελιδοποίηση + Τμηματοποίηση

- Αν ένα τμήμα δε χρησιμοποιείται δε χρειάζεται πίνακας σελίδων για το τμήμα αυτό
- Κοινή μνήμη μεταξύ διεργασιών σε 2 επίπεδα
  - Κοινές σελίδες ή
  - Κοινό τμήμα (κοινός πίνακας σελίδων)
- Η σελιδοποίηση
  - Εξαλείφει τον εξωτερικό κατακερματισμό
  - Επιτρέπει στα τμήματα να μεγαλώνουν
    - Χωρίς να χρειάζεται να μεταφερθούν στη μνήμη
- Για «λογικά» μεγέθη σελίδας, ασήμαντος εσωτερικός κατακερματισμός (.5 σελίδα / τμήμα)
- Πίνακες σελίδων αόρατοι από το χρήστη
  - Γιατί;
  - Ενίοτε μόνο ανάγνωση
- Αν οι πίνακες σελίδων / τμημάτων είναι στη μνήμη, σημαντική επιβάρυνση χρόνου
  - 1 ή 2 «διαχειριστικές» αναφορές στη μνήμη για κάθε πραγματική αναφορά
  - … αλλά και σημαντική επιβάρυνση χώρου για μεγάλα τμήματα

Και ο Νικητής είναι…

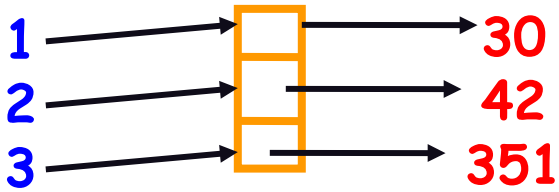
- Απλότητα:
  - Τα περισσότερα συστήματα χρησιμοποιούν σελιδοποίηση
  - Ακόμα και κάποια παλιά που ξεκίνησαν αποκλειστικά με τμηματοποίηση
- Αλλά πολλά συστήματα επιστρέφουν σε «υπερ»-σελίδες (πολύ μεγάλες σελίδες) για την απεικόνιση μεγάλων περιοχών μνήμης
  - Π.χ. hugeTLB στο Linux (σελίδες 16 MB)
  - Γιατί;
    - Πίνακας ακεραίων (32 bits), n x n, μεγέθους 8MB στη μνήμη. Σελιδοποίηση με σελίδες 4K. Εγγραφή κατά στήλες
      - Πόσες σελίδες θα «αγγίξει»;

Τι να Θυμάστε…

- Ιδεατή μνήμη:
  - Ευελιξία + Προστασία + Ταχύτητα (αν γίνει σωστά)
- Βάση + όριο: Απλή απεικόνιση + προστασία
  - + Απλό, γρήγορο
  - - Δύσκαμπτο…
- Τμηματοποίηση με πολλαπλά τμήματα: Γενίκευση του βάση + όριο
  - + Πιο ευέλικτη διαμοίραση μνήμης μεταξύ διεργασιών, καλύτερη χρήση του χώρου
  - - Συνεχής φυσική μνήμη για τα τμήματα
- Σελιδοποίηση: Χρησιμοποίηση «βασικές μονάδες» σταθερού μεγέθους
  - Χρησιμοποίησε πίνακα για την απεικόνιση ιδεατών σελίδων σε φυσικές
  - + Εξαφανίζει τον εξωτερικό κατακερματισμό, Ευέλικτες απεικονίσεις
  - - Εσωτερικός κατακερματισμός, Αναίτια υψηλό κόστος για την απεικόνιση συνεχών περιοχών μνήμης

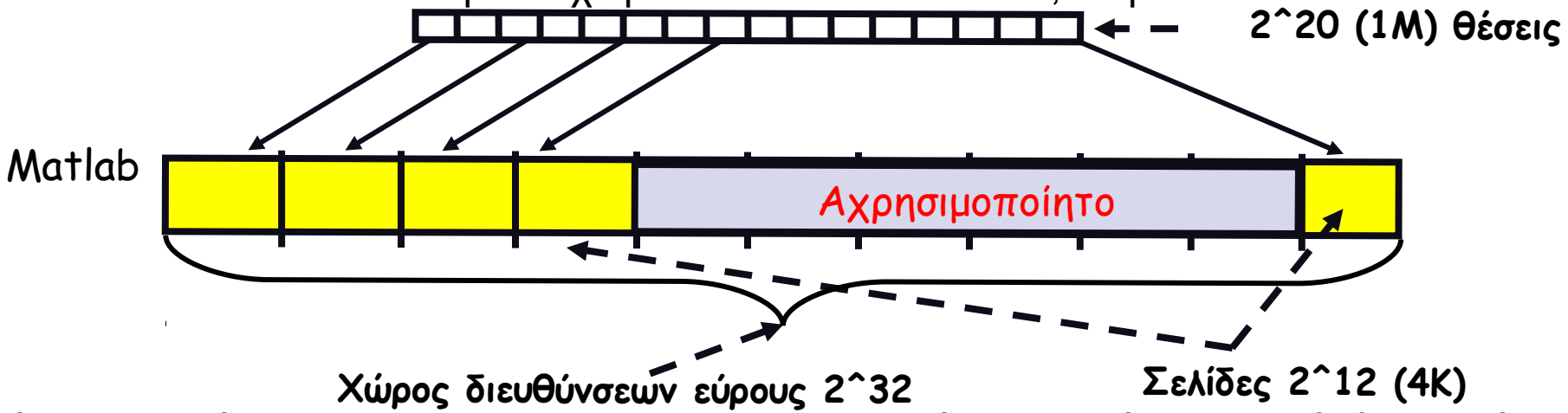
Ιδεατή Μνήμη: Τόσο Πολύπλοκη αλλά … και τόσο Απλή

- Παρά τους φανταχτερούς όρους, τα πάντα είναι συναρτήσεις απεικόνισης
  - Ακεραίων (ιδεατή διεύθυνση) σε ακεραίους (φυσική διεύθυνση)
    - Συμβαίνει και σε άλλες περιοχές του λειτουργικού και της επιστήμης μας…
      - DNS, Filesystems, Μεταβλητές σε καταχωρητές, …
- Το λειτουργικό δεν έχει ένα καλό τρόπο να φτιάξει μια «κομψή» συνάρτηση απεικόνισης
  - Άρα χρησιμοποιούμε πίνακες αναζήτησης (lookup tables)
  - Πίνακας σελίδων / τμημάτων: Πίνακας αναζήτησης…
- Συνήθεις εναλλακτικές
  - Διάνυσμα, hash function, πολυεπίπεδα δέντρα διανυσμάτων
  - Επιλογές μεταξύ κόστους σε χώρο / χρόνο



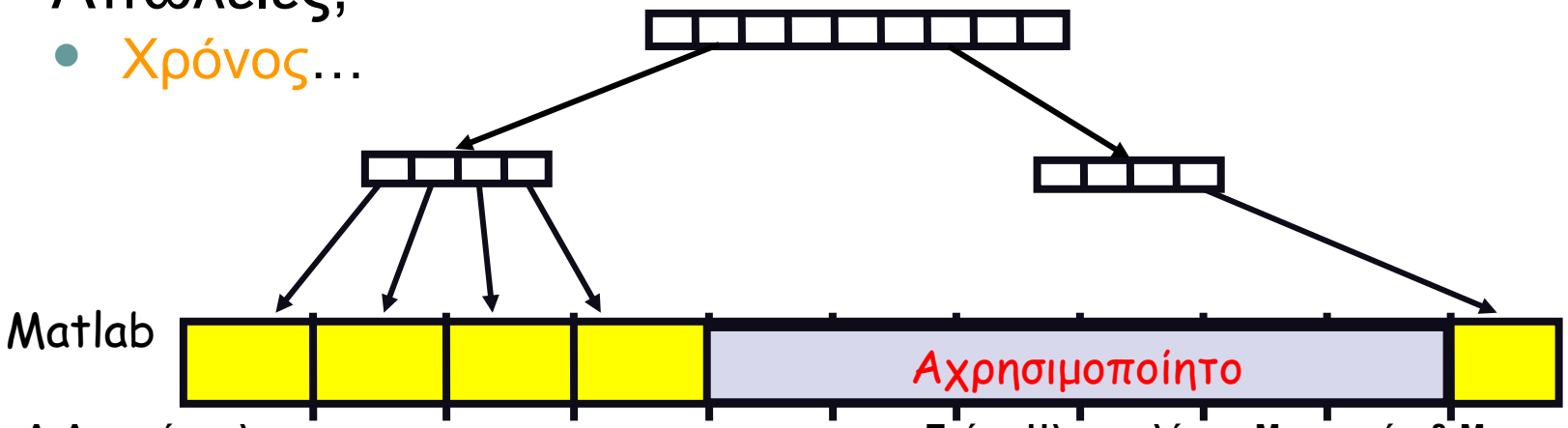
Save our Mem…

- Μεγάλες Περιοχες ⇔ Μεγάλοι Πίνακες Σελίδων…
  - Μεγάλες περιοχές με μικρή χρησιμοποίηση ⇔ μεγάλα αχρησιμοποίητα κομμάτια στον πίνακα σελίδων
    - Πόσο μεγάλη σπατάλη;
      - Χώρος διευθύνσεων 32 bits (4 GB), σελίδες 4KΠόσος χώρος για page table? (αν υποθέσουμε 32 bits ανά θέση)
      - Πόση σπατάλη χώρου αν η χρήση του χώρου διευθύνσεων είναι 10%
      - Και αν πάμε σε χώρο διευθύνσεων 64 bits; Oops…



Ιδέα: Ιεραρχικοί Πίνακες Σελίδων

- Απεικόνισε μικρές περιοχές με πίνακες σελίδων και αυτούς με άλλους πίνακες σελίδων και αυτούς … κ.ο.κ.
- Κέρδος;
  - Χώρος: Δε φτιάχνουμε πίνακες σελίδων για αχρησιμοποίητα τμήματα του χώρου διευθύνσεων
    - Πάντα κερδίζουμε σε χώρο; Καμιά «κακή» περίπτωση;
- Απώλειες;
  - Χρόνος…



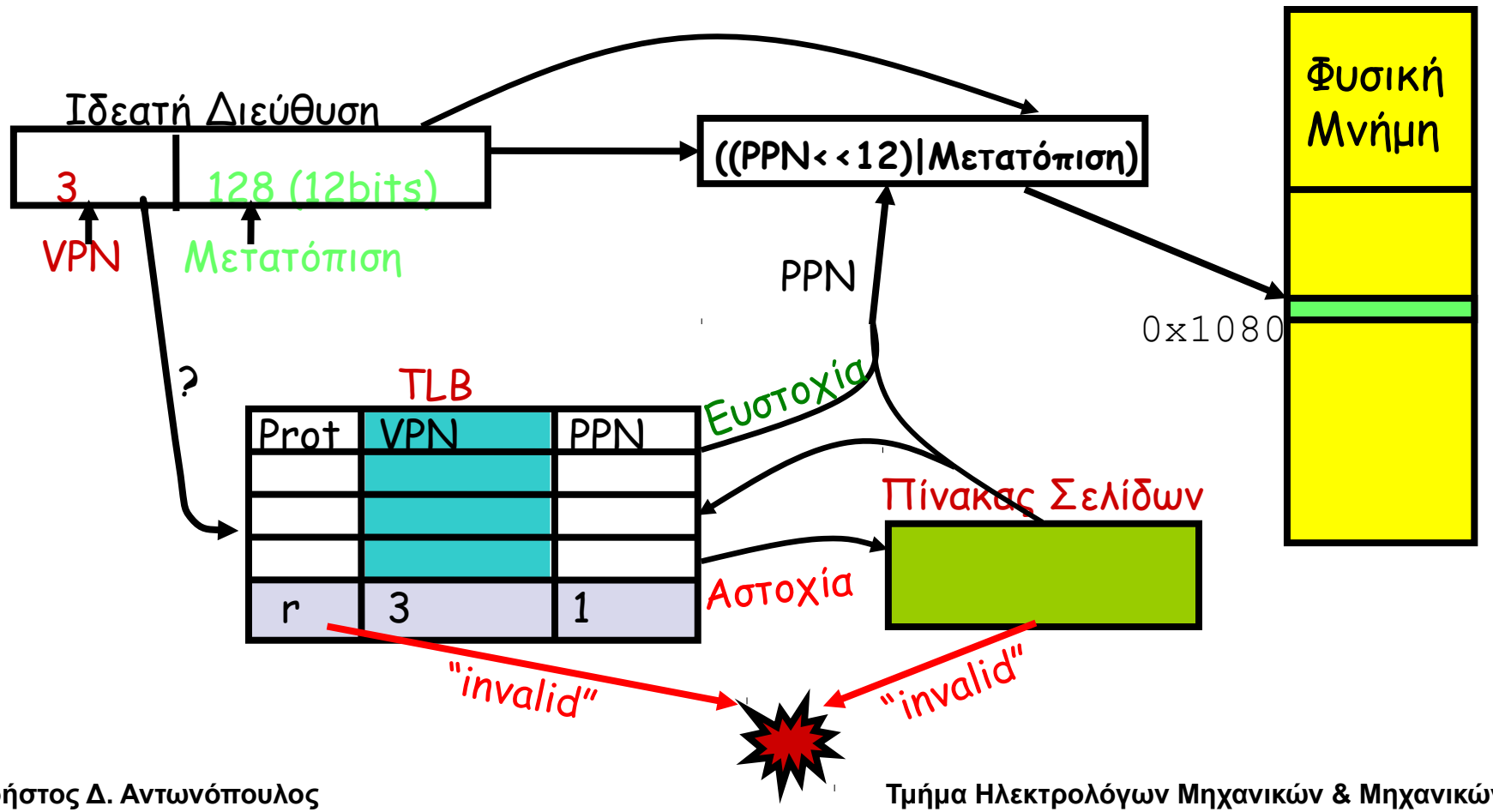
Το Μεγάλο Πρόβλημα: Επίδοση

- Για κάθε πρόσβαση από το πρόγραμμα στη μνήμη…
  - … μία ή και (αρκετά) περισσότερες προσβάσεις στον/στους πίνακα/ες σελίδων (στη μνήμη)
- Η προφανής ιδέα;
  - Caching!!!
  - Τι;
    - Μεταφράσεις ιδεατών σελίδων σε φυσικά πλαίσια
  - Μα πώς;
    - Το πρόγραμμα κατά κανόνα δε χρησιμοποιεί συνεχώς όλο το χώρο διευθύνσεων του…
    - … κάθε χρονική περίοδο περιορίζεται σε ένα συγκεκριμένο υποσύνολο του χώρου διευθύνσεων …
    - … και το υποσύνολο αυτό αλλάζει συνήθως αργά …
  - Άρα
    - Θυμήσου τις πιο πρόσφατες μεταφράσεις ιδεατών σελίδων σε φυσικά πλαίσια

Ο TLB (Translation Look-Aside Buffer)

- Τι είναι;
  - Πολύ γρήγορη / πολύ μικρή μνήμη
  - Μπορείς να ψάξεις όλες τις θέσεις της ταυτόχρονα
- Τι περιέχει;
  - Μετάφραση ιδεατής σελίδας σε φυσικό πλαίσιο + πληροφορία προστασίας
- Σε κάθε αναφορά στη μνήμη:
  - Κοίτα στον TLB, αν την βρεις … μεγάλη ταχύτητα.
  - Αν όχι, κοίτα στους πίνακες σελίδων, βάλε τη μετάφραση στον TLB για την επόμενη φορά
    - Μπορεί να χρειαστεί να πετάξεις τα περιεχόμενα κάποιας θέσης αν είναι γεμάτος…
      - Ποιάς;
  - Κατά κανόνα:
    - 16-2K θέσεις, ευστοχία 95%

Ένα Σύστημα με TLB



TLB: Λεπτομέρειες

- Ταχύτητα: Ταχύτητα της «καρδιάς» του επεξεργαστή
- Πρόβλημα:
  - Τι κάνουμε όταν αλλάζουμε χώρους διευθύνσεων; (διεργασίες)
    - Καθάρισε τον TLB σε κάθε μεταγωγή περιβάλλοντος (π.χ. Intel x86)
    - Μάρκαρε κάθε θέση με το αναγνωριστικό της διεργασίας στην οποία «ανήκει» αυτή η θέση (π.χ. MIPS)
    - Γενικά, το ΛΣ πρέπει να φροντίζει το ίδιο να διατηρεί τον TLB σε «νόμιμη» κατάσταση
    - Άλλη περίπτωση: Αν αλλάξει μια εγγραφή στον πίνακα σελίδων
- Φόρτωση του TLB μετά από αστοχία
  - Από το υλικό ή από το λογισμικό (ΛΣ)
    - Υλικό: Ψάξε στους πίνακες σελίδων (Intel x86)
    - Λογισμικό: Στείλε μήνυμα στο ΛΣ να το φροντίσει…
      - Πλεονέκτημα: Μόνο το ΛΣ ακουμπά τους πίνακες σελίδων…
      - … και άρα μπορεί να αποφασίσει για τη μορφή τους

Πού / Πώς είναι το ΛΣ στη Μνήμη;

- Κατά κανόνα:
  - Ο χώρος ιδεατών διευθύνσεων κάθε διεργασίας περιλαμβάνει και τη μνήμη που είναι διαθέσιμη σε επίπεδο χρήστη, και αυτή που ανήκει στο ΛΣ.
  - Το ΛΣ βρίσκεται στην ίδια περιοχή ιδεατής μνήμης σε κάθε διεργασία
    - Π.χ. MIPS: Τα πάνω 2 GB του ιδεατού χώρου διευθύνσεων για το ΛΣ.
    - Π.χ. Linux, IA32; Το πάνω 1 GB του ιδεατού χώρου διευθύνσεων για το λειτουργικό
  - Βελτίωση: Το ΛΣ τρέχει χωρίς «μετάφραση» ιδεατών/φυσικών διευθύνσεων
    - Περιοχή ιδεατών διευθύνσεων που όταν τρέχουμε σε επίπεδο ΛΣ απεικονίζεται στη φυσική μνήμη με την αφαίρεση μιας σταθεράς (κάτι σαν το 1 τμήμα)
- Τι γίνεται αν μια εφαρμογή πάει να διαβάσει / γράψει μνήμη του ΛΣ; Πώς πετυχαίνουμε την προστασία;

Προσπέλαση Δεδομένων Χρήστη από το ΛΣ - Προβλήματα

- Γιατί;
  - Π.χ. Κλήσεις συστήματος I/O πρέπει να επιστρέψουν δεδομένα σε περιοχές μνήμης που έχει καθορίσει ο χρήστης
- Πώς;
  - Μπορούμε να εμπιστευθούμε διευθύνσεις που μας δίνει ο χρήστης;
    - Ο χρήστης δίνει NULL η μια προφανώς λανθασμένη διεύθυνση (π.χ. 0xffffff00)
    - Λιγότερο προφανές: Ο χρήστης δίνει μια ιδεατή διεύθυνση που ανήκει στον πυρήνα
  - Η περιοχή μνήμης που έδωσε ο χρήστης μπορεί να βρίσκεται – τη στιγμή αυτή – στο δίσκο
    - Σφάλμα σελίδας στη μέση μιας κλήσης συστήματος
    - Αν δοθεί ο επεξεργαστής σε άλλη διεργασία… (μαύρο φίδι που μας έφαγε…)
      - Αν η διεργασία που «βγάλαμε» είχε κάποιο lock του ΛΣ;
      - Αν η κλήση συστήματος είχε αλλάξει κάποιες δομές του πυρήνα, αλλά δεν είχε προλάβει να τις φέρει όλες σε «νόμιμη» κατάσταση;
  - Μπορούμε να μεταφράσουμε αυτόματα ιδεατές διευθύνσεις που μας δίνει ο χρήστης;

Σύνοψη: Απεικόνιση Ιδεατής Μνήμης

- Τι είναι;
  - Δώσε στον προγραμματιστή λογικές (ιδεατές) διευθύνσεις αντί των πραγματικών (φυσικών) για να παίζει…
- Πώς γίνεται;
  - Μετάφρασε κάθε προσπέλαση μνήμης χρησιμοποιώντας έναν πίνακα (πίνακας σελίδων, πίνακας τμημάτων).
  - Ταχύτητα: Αποθήκευσε σε γρήγορη μνήμη συχνά χρησιμοποιούμενες μεταφράσεις
- Τι πετύχαμε;
  - Κάθε διεργασία έχει ένα δικό της χώρο διευθύνσεων
  - Οι διεργασίες τρέχουν με τον ίδιο τρόπο, ανεξάρτητα από το που έχουν τοποθετηθεί στη φυσική μνήμη και πόσο μεγάλη είναι η φυσική μνήμη
  - Προστασία: Διασφαλίζουμε ότι κάθε διεργασία «ακουμπά» μόνο τη δική της μνήμη.
- Στο επόμενο μάθημα…
  - Οι σελίδες-γυρολόγοι, ή … πώς να κάνετε τη μνήμη σας να συμπεριφέρεται σαν να είναι ακόμα μεγαλύτερη