# Λειτουργικά Συστήματα (HY222)

Διάλεξη 7:

Εισαγωγή στην Ιδεατή Μνήμη - Τμηματοποίηση

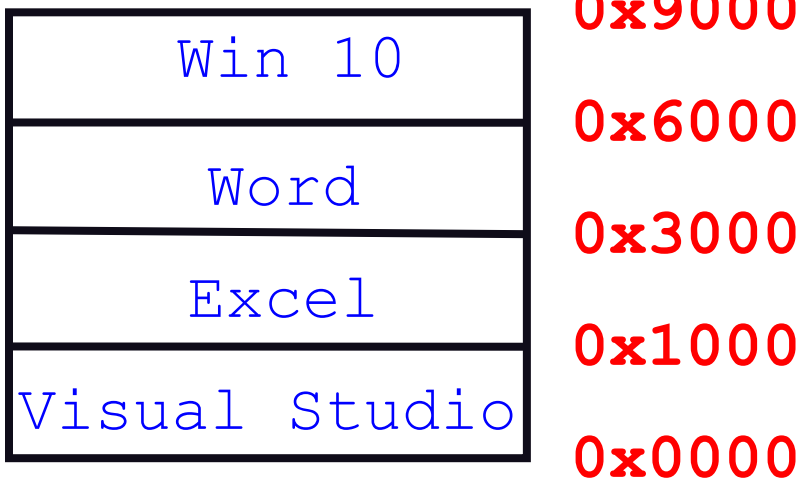
**Χρήστος Δ. Αντωνόπουλος**
**23/3/2017**

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

# Η Ευτυχισμένη Κοινωνία των Διεργασιών

- Πολλαπλές χαρούμενες διεργασίες στο σύστημα

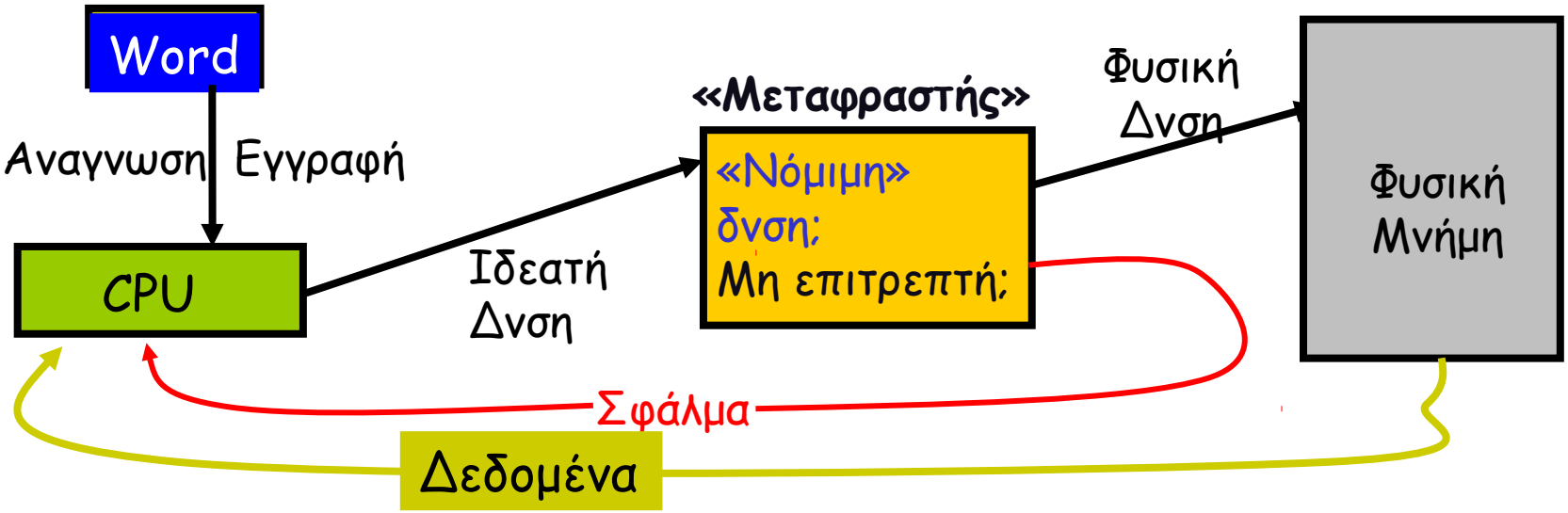
- Και αν το Visual Studio χρειαστεί κι άλλη μνήμη;
- Αν το Excel χρειαστεί περισσότερη μνήμη από τη διαθέσιμη στη μηχανή; (τι σπάνιο…)
- Αν το Word πάει να γράψει στη διεύθυνση 0x634; (τι σπάνιο…)
- Αν το Word αποφασίσει να αφήσει λίγη από τη μνήμη του;
- Και πότε θα μάθει το Visual Studio ότι ξεκινάει από τη διεύθυνση 0x0000;



# Απαιτήσεις για τη Διαμοίραση Κοινής Μνήμης

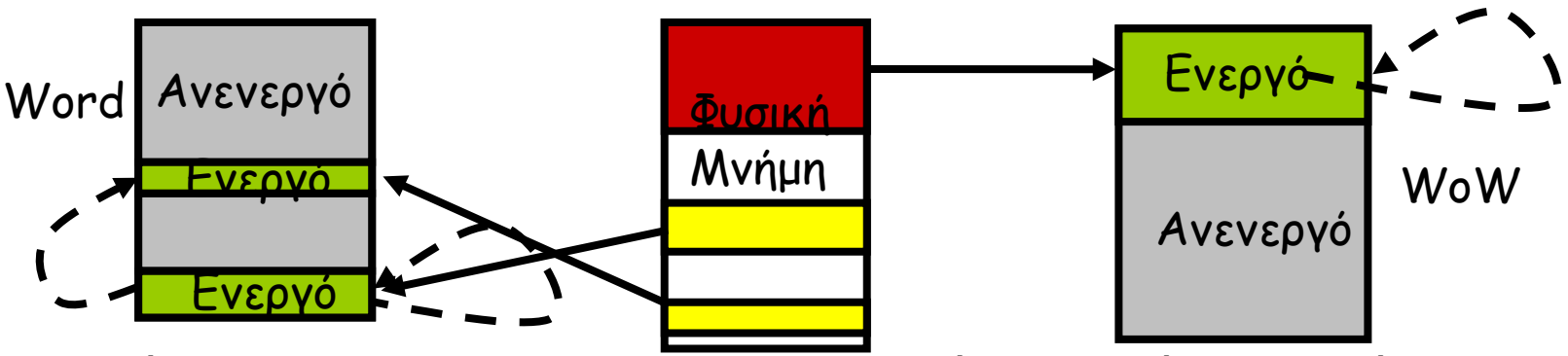
- Προστασία: Αμαρτίες διεργασιών παιδεύουσι μόνο τις ίδιες
  - Έλεγξε κάθε προσπέλαση στη μνήμη. Δες αν είναι μέσα στα «επιτρεπτά» όρια.
- Διαφάνεια: Ας μπορεί η διεργασία να τοποθετηθεί οπουδήποτε στη μνήμη
  - Αξίωμα Roger Needham: Οποιοδήποτε πρόβλημα της επιστήμης των υπολογιστών μπορεί να λυθεί προσθέτοντας ένα ακόμη επίπεδο ανακατεύθυνσης…
    - Ακόμα καλύτερα αν το κάνει κάποιος άλλος (π.χ. ο σχεδιαστής του επεξεργαστή)
  - Δώσε της την «ψευδαίσθηση» μιας μεγάλης, ιδιόκτητης μνήμης. Κατά την εκτέλεση μετέφρασε τις προσπελάσεις στην «ιδιόκτητη» μνήμη σε προσπελάσεις πραγματικής μνήμης.

# Το Μεγάλο Κόλπο



# Μ’ ένα σμπάρο…

- Ευελιξία ΚΑΙ αποδοτικότητα!
- Ιδεατή μνήμη: Επίπεδο ανακατεύθυνσης μεταξύ εφαρμογών και πραγματικής μνήμης
  - Ευελιξία: Οι διεργασίες μπορούν να τοποθετηθούν οπουδήποτε στη μνήμη…
  - Απλότητα: … και ο προγραμματιστής δε χρειάζεται να ξέρει τίποτα
  - Αποδοτικότητα: Η περισσότερη ιδεατή μνήμη δε θα χρησιμοποιείται (ο χρυσός κανόνας του 80/20)
    - Άσε άλλες διεργασίες να χρησιμοποιούν τα «ανενεργά» κομμάτια.
    - Κάνε τη μνήμη να φαίνεται πολύ μεγαλύτερη!
- Προβλήματα:
  - Ένα επιπλέον επίπεδο ανακατεύθυνσης
  - Χωρίς προσοχή… καλή υπομονή (αργόοοοο)

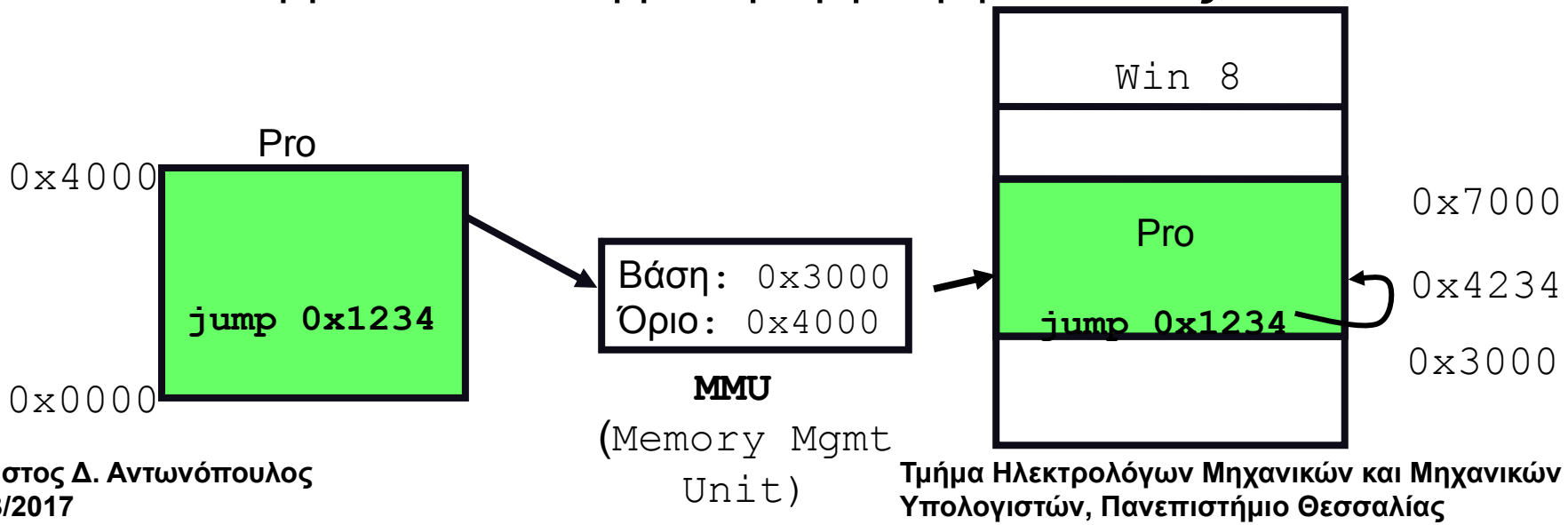


# Για κάθε λύση…

- … και ένα πρόβλημα
  - Πώς επιβάλλεται η προστασία;
  - Πώς κομματιάζεται η μνήμη;
  - Πώς τοποθετείται η διεργασία στη μνήμη;
    - Όταν μεγαλώσετε… (αυστηρώς ακατάλληλο για ανυποψίαστους μηχανικούς)
    - … αν έχουμε χρόνο στο τέλος του εξαμήνου

# Το Αυγό του Κολόμβου - Τμηματοποίηση: Βάση + Όριο

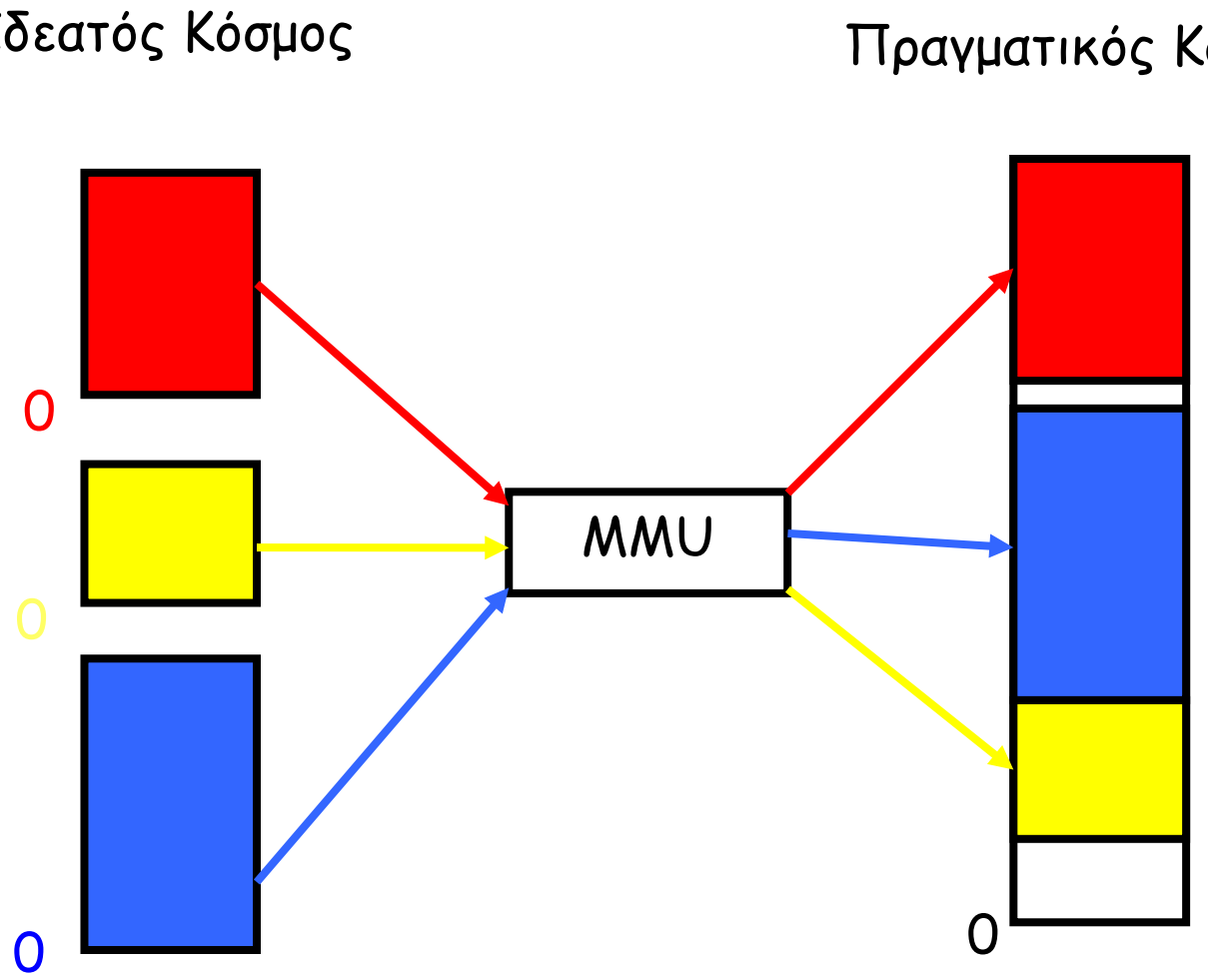
- Βάλε δουλειά στο σχεδιαστή του υλικού:
  - Σε κάθε προσπέλαση μνήμης
    - Φυσική Δνση = Ιδεατή Δνση + Βάση (μετατόπιση)
    - ΠΡΕΠΕΙ Φυσική Δνση στο [Βάση, Βάση + Όριο)
- Ερωτήσεις:
  - Πώς μετακινούμε διεργασίες στη μνήμη;
  - Τι συμβαίνει κατά τη μεταγωγή περιβάλλοντος;



# Λίγη «Αργκό»

- Διευθύνσεις στο πρόγραμμα: Ιδεατές ή λογικές
- Αληθινές διευθύνσεις (στον κόσμο της σιλικόνης): Πραγματικές ή φυσικές
- Μηχανισμός μετάφρασης ή επανατοποθέτησης:
  - Μονάδα διαχείρισης μνήμης (Memory management Unit – MMU)
  - Ίδια λογική για όλους τους μηχανισμούς δυναμικής μετάφρασης (δείτε «το μεγάλο κόλπο»)
    - Πολλαπλές διαφορετικές εικόνες (ψευδαισθήσεις) της μνήμης: Χώροι διευθύνσεων

# Ιδεατός vs. Πραγματικός Κόσμος

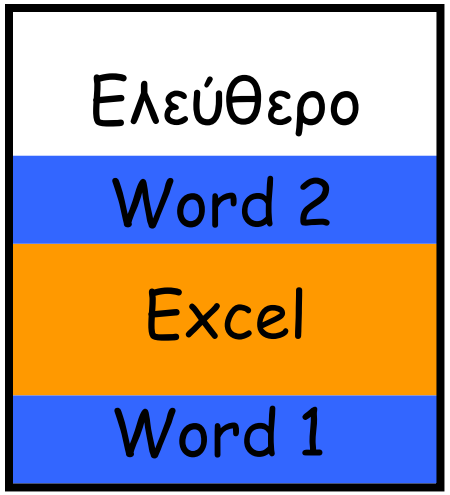


# Μηχανισμοί Προστασίας

- Και πώς θα σταματήσω το χρήστη από το να αλλάξει τη βάση και το όριο;
  - Ειδικοί καταχωρητές για τη βάση και το όριο
  - Προνομιούχες εντολές για την αλλαγή τους
- Και πώς μεταφερόμαστε;
  - Από τις διεργασίες χρήστη στο λειτουργικό;
    - Στο λειτουργικό έχουμε τον ίδιο χώρο διευθύνσεων για όλες τις διεργασίες
  - Από μία διεργασία χρήστη σε άλλη;

# Η ώρα της αλήθειας

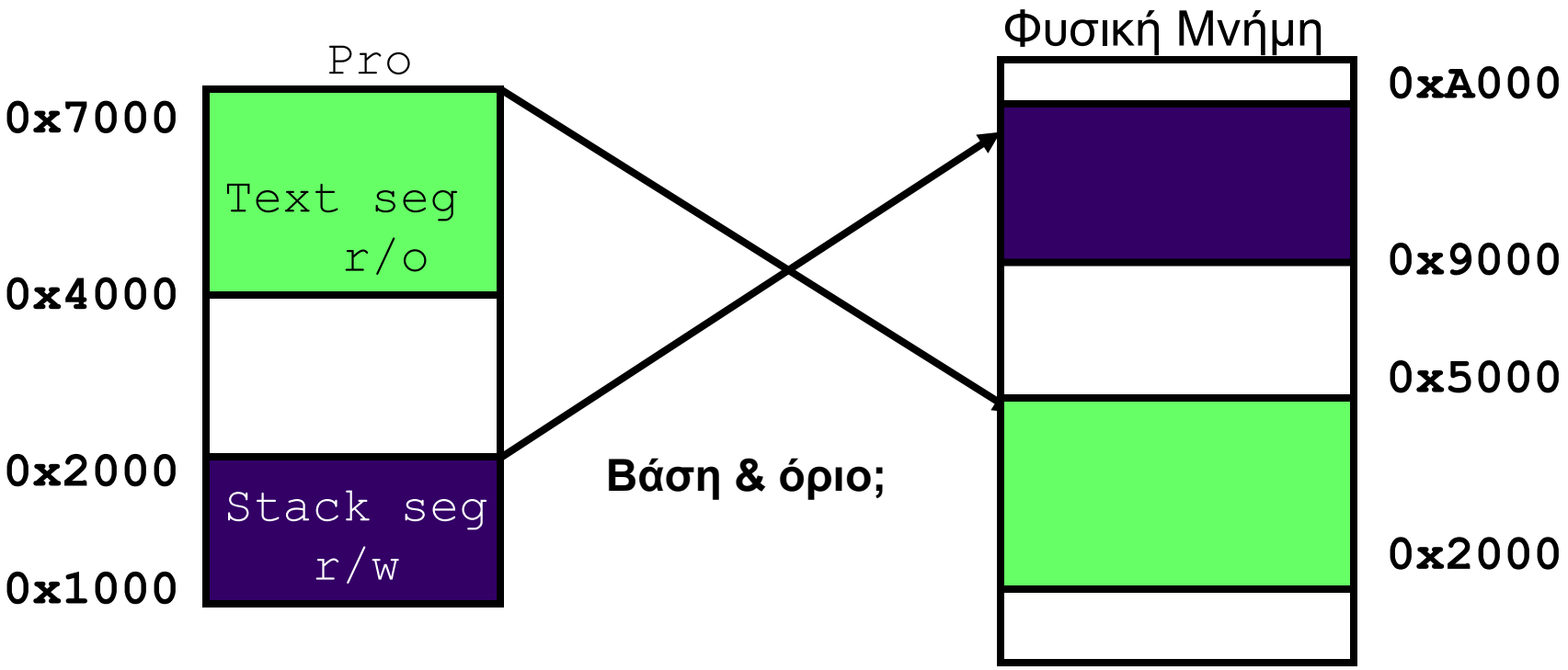
- Πλεονεκτήματα:
  - Φθηνό: Μόνο 2 καταχωρητές
  - Γρήγορο: Πρόσθεση και σύγκριση
    - Σε κάποια συστήματα με 1 εντολή
- Μειονεκτήματα:
  - Διεργασίες που «μεγαλώνουν» (π.χ. Excel)
  - Πώς να μοιραστούμε μνήμη; (π.χ. τον κώδικα του Word)
    - Και πώς ξεχωρίζουμε κώδικα και δεδομένα;
- Λύση;
  - Πολλαπλά τμήματα – Τμηματοποίηση (segmentation)



# Τμηματοποίηση με Πολλαπλά Τμήματα (Segmentation)

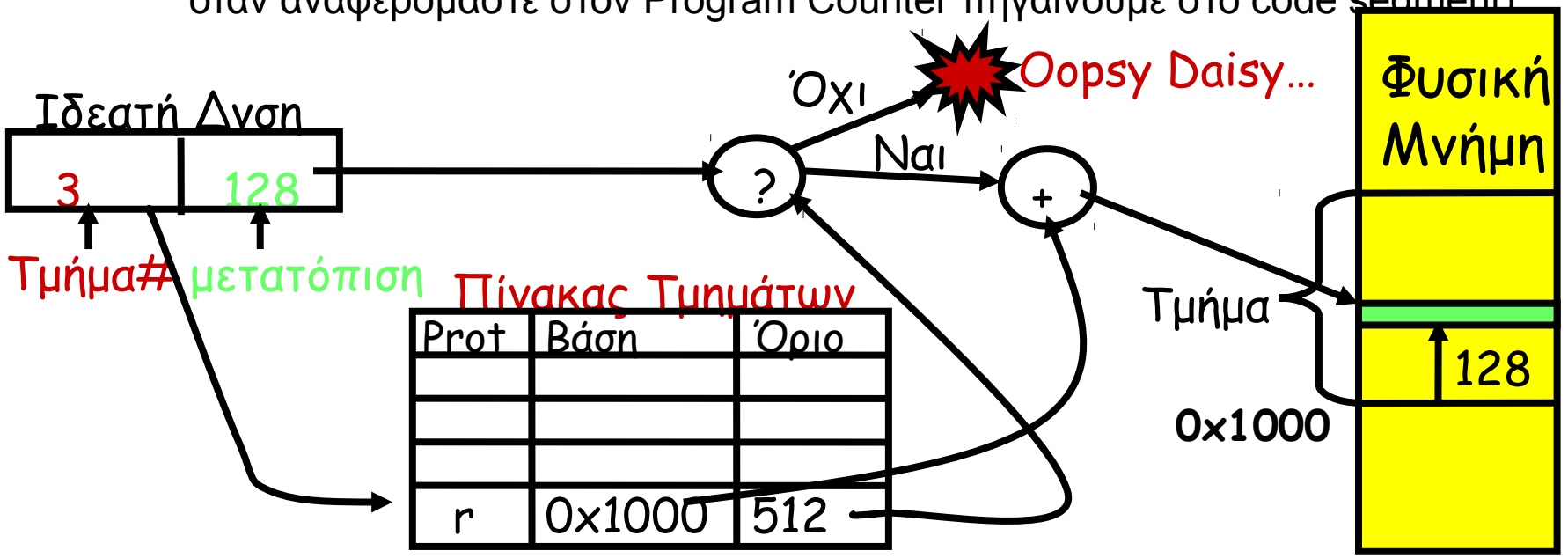
- Ιδέα: 
  - Άσε τις διεργασίες να έχουν περισσότερα από ένα τμήματα (βάσεις και όρια)
    - Ο χώρος διευθύνσεων «χτίζεται» από αυτά τα πολλαπλά τμήματα
    - Κάθε διεργασία έχει τις δικές τις τιμές βάσεων και ορίων
    - Τώρα μπορούμε να έχουμε κοινά τμήματα μνήμης
      - Πώς;
    - Αφού μπορούμε να έχουμε κοινά τμήματα μνήμης, πρέπει να υλοποιήσουμε δικαιώματα
      - Read / Write bits (σε κάποια συστήματα και Execute bit)
- Αποτέλεσμα:
  - Ο χώρος διευθύνσεων κάθε διεργασίας μπορεί να είναι «διασκορπισμένος» στη φυσική μνήμη
- Πρόβλημα:
  - Και που ξέρουμε σε ποιο τμήμα ανήκει η κάθε διεύθυνση;

# Τμηματοποίηση: Παράδειγμα



# Μηχανισμοί Τμηματοποίησης

- Κάθε διεργασία έχει το δικό της πίνακα τμημάτων
- Κάθε αναφορά στη μνήμη καθορίζει τμήμα και μετατόπιση μέσα στο τμήμα
  - Χρησιμοποιούνται κάποια bits της διεύθυνσης για τον καθορισμό του τμήματος
  - Το τμήμα επιλέγεται έμμεσα, ανάλογα με την εντολή ή τα έντελά της (π.χ. όταν αναφερόμαστε στον Program Counter πηγαίνουμε στο code segment)

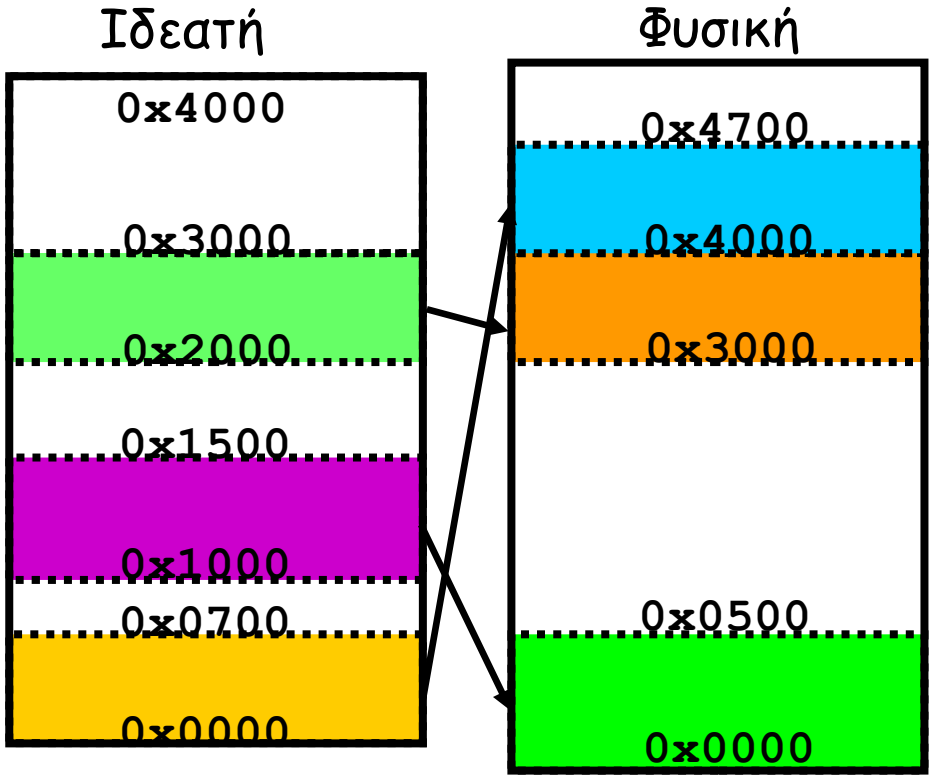


# Τμηματοποίηση: Παράδειγμα

- 2 bits (περισσότερο σημαντικά) αριθμός τμήματος, 12 bits μετατόπιση

| Τμ. | Βάση | Όριο | rw |
|---|---|---|---|
| 0 | 0x4000 | 0x6ff | 10 |
| 1 | 0x0000 | 0x4ff | 11 |
| 2 | 0x3000 | 0xfff | 11 |
| 3 | | | 00 |

- Και… κατά που πέφτει το…
  - 0x0240;
  - 0x1108;
  - 0x265c;
  - 0x3002;
  - 0x1600;



# Τμηματοποίηση: Υπέρ & Κατά

- Υπέρ:
  - Πολλαπλά τμήματα για κάθε διεργασία
  - Επιτρέπεται διαμοίραση τμημάτων του χώρου διευθύνσεων μεταξύ διεργασιών
  - Δε χρειάζεται να βρίσκεται όλη η διεργασία (όλος ο χώρος διευθύνσεων της) στη μνήμη!!!
    - Γιατί είναι αυτό σημαντικό;
- Κατά:
  - Ένα ακόμα επίπεδο ανακατεύθυνσης
    - Χρειαζόμαστε υποστήριξη από το υλικό για ταχύτητα
  - Για τα n bytes ενός τμήματος χρειαζόμαστε n συνεχόμενα bytes στη φυσική μνήμη
    - Γιατί;
    - Πρόβλημα;
      - «Κατακερματισμός» της φυσικής μνήμης

# Κατακερματισμός Μνήμης

- Αδυναμία χρήσης μνήμης που είναι ελεύθερη
  - «Κομμάτια» μεταβλητού μεγέθους: Πολλές μικρές τρύπες στη φυσική μνήμη (εξωτερικός κατακερματισμός)
  - «Κομμάτια» σταθερού μεγέθους: Δεν έχουμε τρύπες στη φυσική μνήμη. Έχουμε όμως τρύπες (αχρησιμοποίητη μνήμη) εσωτερικά σε κάθε κομμάτι (εσωτερικός κατακερματισμός)