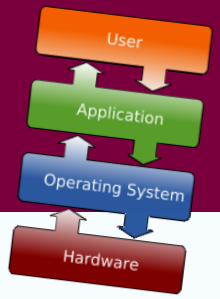


Ιδεατή Μνήμη

Ανδρέας Λ. Συμεωνίδης

Αν. Καθηγητής

Τμήμα Ηλεκτρολόγων Μηχ/κών

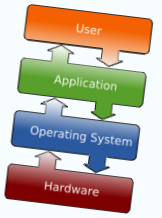
&

Μηχ/κών Υπολογιστών, Α.Π.Θ.

Email:asymeon@eng.auth.gr

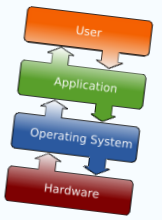
Στόχοι της Δ-8

- Να ολοκληρώσει τη μελέτη πάνω στη διαχείριση μνήμης
- Να παρουσιάσει θέματα σύνδεσης και φόρτωσης της μνήμης
- Να διαχωρίσει ανάμεσα σε ιδεατές και πραγματικές διευθύνσεις
- Να επεξηγήσει τη μετάφραση της λογικής σε σχετική και φυσική διεύθυνση
- Να παρουσιάσει τις τεχνικές τμηματοποίησης της ιδεατής μνήμης
  - Σελιδοποίηση
  - Πίνακες σελίδων
  - Κατάτμηση
  - Πίνακες τμημάτων
  - Κατάτμηση με σελιδοποίηση
- Να δώσει μια σειρά από ασκήσεις πάνω στην ιδεατή μνήμη



Αντί ανακεφαλαίωσης…

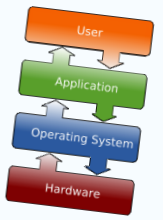
- Η κύρια μνήμη είναι, μετά από το χρόνο χρήσης της CPU, ο δεύτερος πιο σημαντικός πόρος σε ένα υπολογιστικό σύστημα.
- Ακόμη και με σχετικά μεγάλο μέγεθος, η ποσότητα της διαθέσιμης κύριας μνήμης συχνά δεν είναι ικανοποιητική.
- Η λήψη πληροφοριών από τον σκληρό δίσκο αντί της κύριας μνήμης καθυστερεί υπέρμετρα το σύστημα:
  - 60 ns χρόνος προσπέλασης της κύριας μνήμης
  - 10 ms (= 10.000.000 ns) μέσος χρόνος προσπέλασης των σκληρών δίσκων
- Πολλές διεργασίες πρέπει να συνυπάρχουν στη μνήμη



Αντί ανακεφαλαίωσης (συν.)...

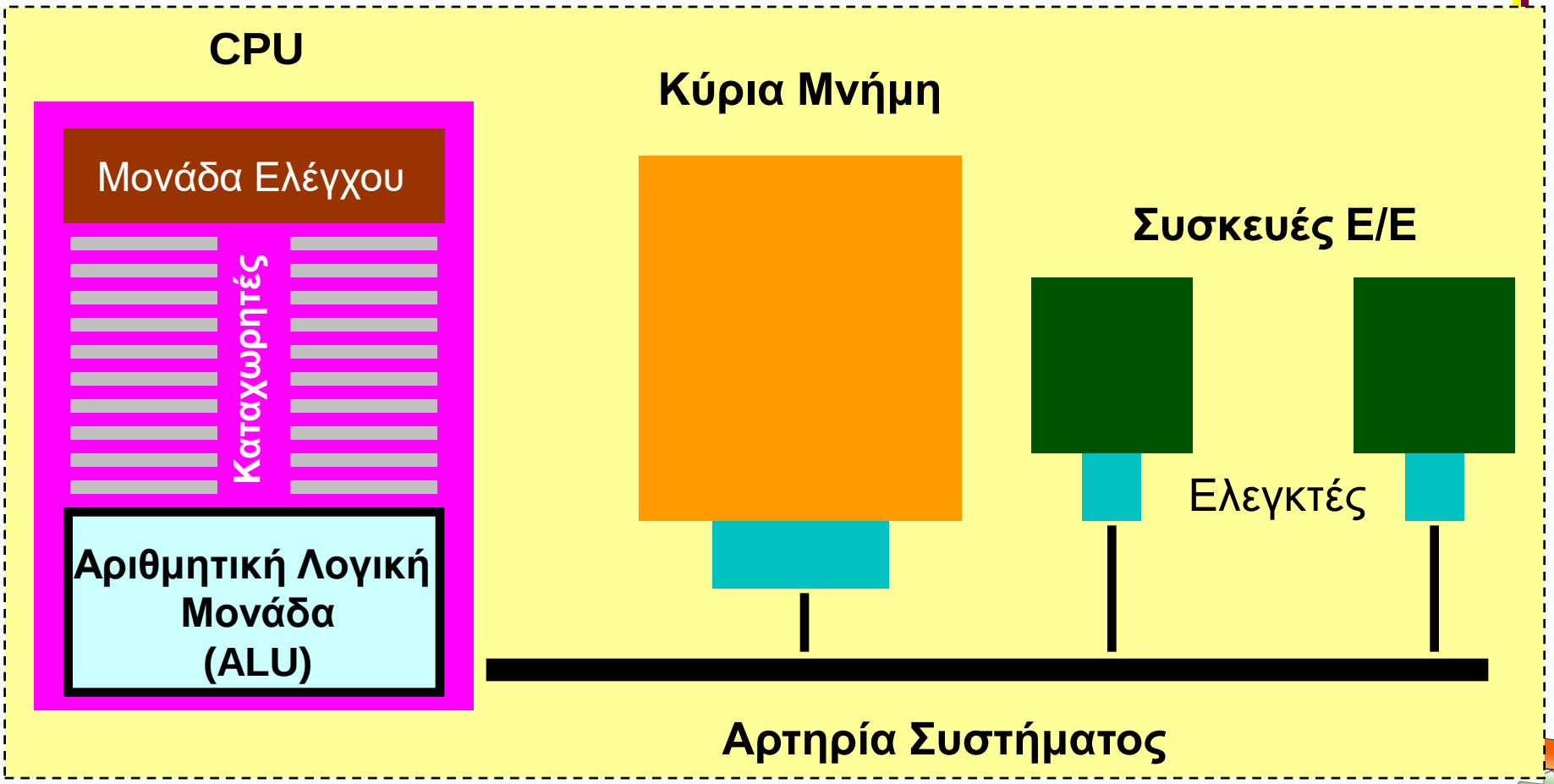
- Η διαχείριση μνήμης επιτυγχάνεται μέσω μιας πολύπλοκης σχέσης μεταξύ του υλικού μέρους του επεξεργαστή και του λογισμικού του ΛΣ
- Οι βασικές τεχνικές διαχείρισης μνήμης ανταγωνίζονται για τη δέσμευση περιορισμένου χώρου στην κύρια μνήμη.
- Η λύση της μεγαλύτερης κύριας μνήμης είναι συνήθως απαγορευτικά δαπανηρή.
- Η δεύτερη λύση είναι η δημιουργία της ψευδαίσθησης ότι υπάρχει περισσότερη μνήμη από όση είναι εγκατεστημένη.

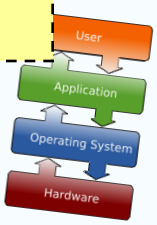
Αυτό αποτελεί τη βασική ιδέα της ιδεατής μνήμης (virtual memory).



A gentle reminder…

Αρχιτεκτονική συστημάτων

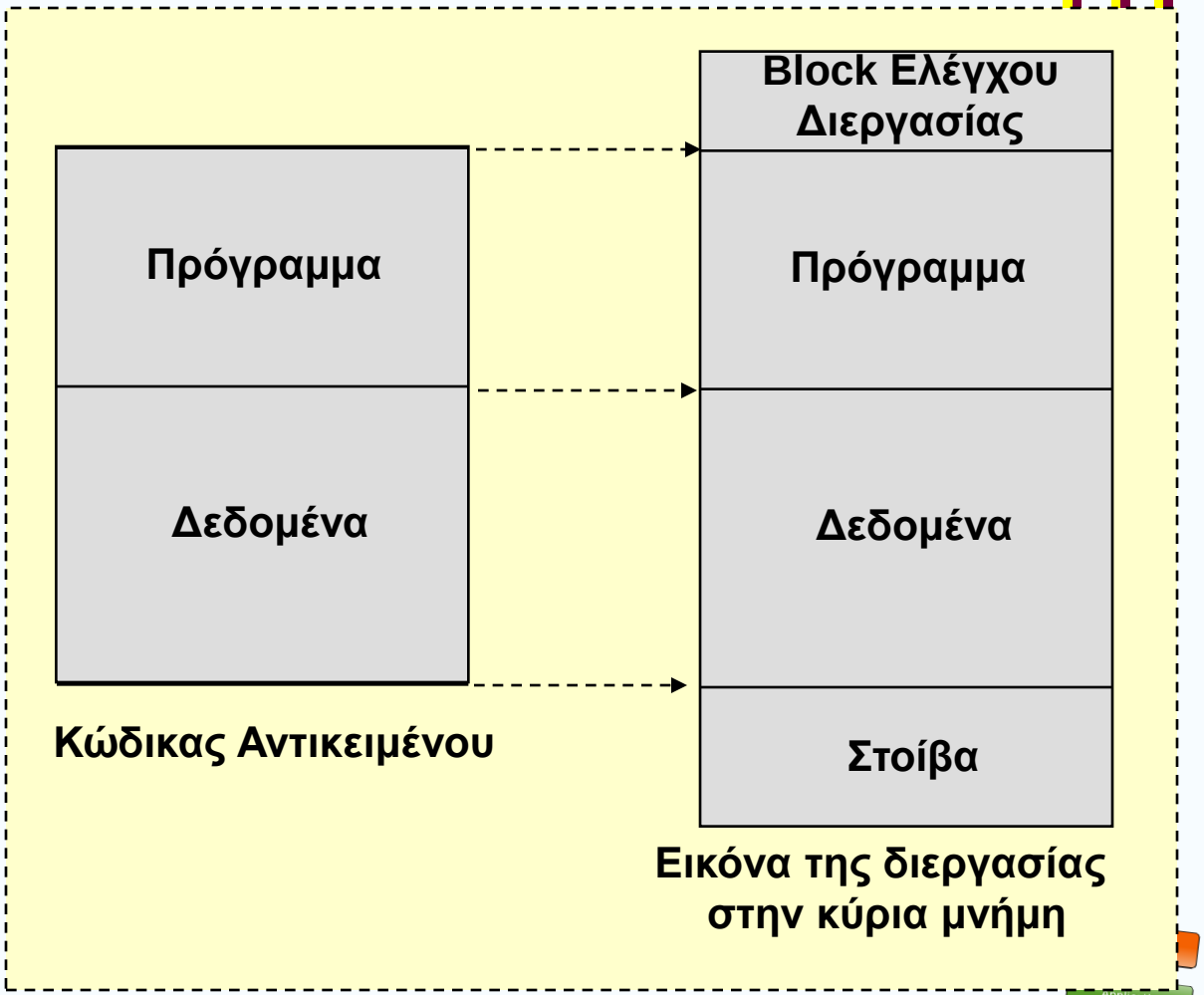


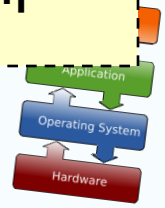


A gentle reminder...

Σύνδεση και Φόρτωση

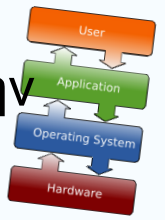
Το πρώτο βήμα κατά τη δημιουργία μιας ενεργής διεργασίας είναι η φόρτωση ενός προγράμματος στην κύρια μνήμη και η δημιουργία της εικόνας διεργασίας.



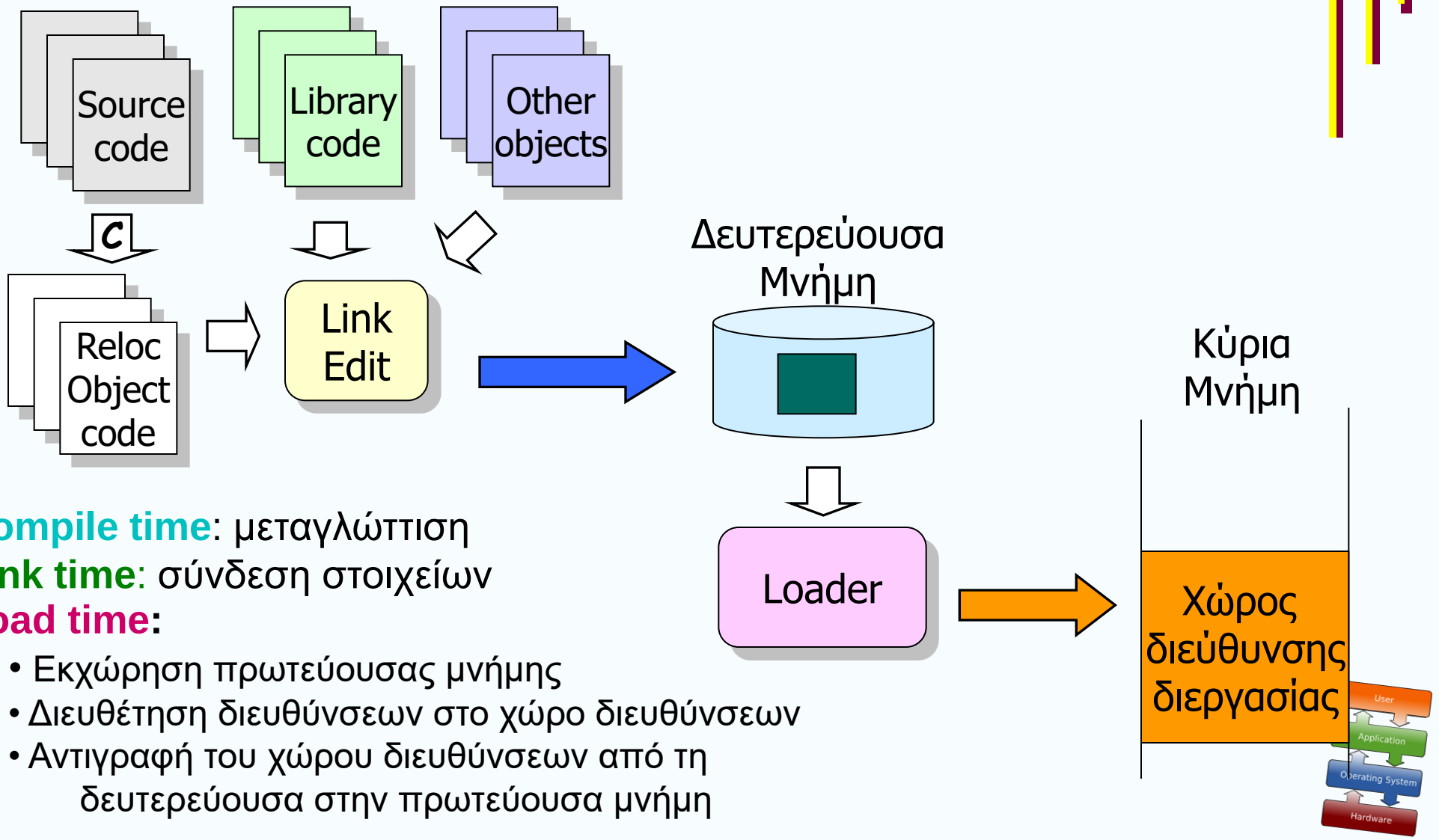


Σύνδεση και Φόρτωση (συν.)

- Κάθε εφαρμογή αποτελείται από έναν αριθμό μεταφρασμένων και μεταγλωττισμένων ενοτήτων (**modules**) με τη μορφή κώδικα αντικειμένου (**object code**).
- Αυτές οι ενότητες συνδέονται για την επίλυση οποιασδήποτε αναφοράς μεταξύ τους, ενώ συγχρόνως πραγματοποιούνται αναφορές και σε ρουτίνες βιβλιοθήκης.
- Οι ρουτίνες βιβλιοθήκης μπορούν να ενσωματωθούν στο πρόγραμμα ή να αναφέρονται ως διαμοιραζόμενος κώδικας που πρέπει να παρέχεται από το ΛΣ κατά τη στιγμή της εκτέλεσης.
- Η μονάδα σύνδεσης (**linker**) παίρνει ως είσοδο μια συλλογή από **modules αντικειμένων** και παράγει ένα **module φόρτωσης** που αποτελείται από ένα ολοκληρωμένο σύνολο από προγράμματα και δεδομένα που θα περάσουν στον φορτωτή. Σε κάθε module αντικειμένου μπορούν να υπάρχουν αναφορές διεύθυνσης σε θέσεις άλλων modules.
- Η μονάδα φόρτωσης τοποθετεί το module που πρόκειται να φορτωθεί στην κύρια μνήμη.



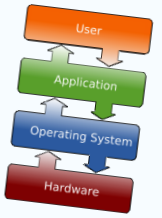
Δόμηση του χώρου διευθύνσεων



- **Compile time**: μεταγλώττιση
- **Link time**: σύνδεση στοιχείων
- **Load time**:
  - Εκχώρηση πρωτεύουσας μνήμης
  - Διευθέτηση διευθύνσεων στο χώρο διευθύνσεων
  - Αντιγραφή του χώρου διευθύνσεων από τη δευτερεύουσα στην πρωτεύουσα μνήμη

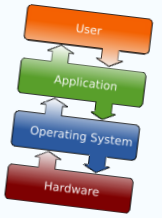
Φόρτωση

- **Απόλυτη φόρτωση**
  - Το υπό φόρτωση module φορτώνεται πάντοτε στην ίδια θέση στην κύρια μνήμη. Οι αναφορές διεύθυνσης θα πρέπει να είναι απόλυτες διευθύνσεις της κύριας μνήμης.
- **Φόρτωση με μετατόπιση (relocation)**
  - Το υπό φόρτωση module μπορεί να τοποθετηθεί οπουδήποτε στην κύρια μνήμη. Ο interpreter ή ο compiler παράγουν διευθύνσεις σχετικές με ένα γνωστό σημείο που είναι συνήθως η αρχή του προγράμματος.
- **Δυναμική εκτέλεση φόρτωσης**
  - Το υπό φόρτωση module φορτώνεται στην κύρια μνήμη με όλες τις αναφορές μνήμης σε σχετική μορφή. Οι απόλυτες διευθύνσεις υπολογίζονται κατά τη στιγμή εκτέλεσης μιας εντολής.



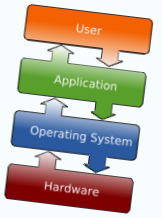
Στρατηγικές σύνδεσης

- Στατική σύνδεση (static linking)
  - Τα αντικείμενα (object files) και οι βιβλιοθήκες συνδυάζονται σε ένα μοναδικό εκτελέσιμο αρχείο κατά τη διάρκεια της σύνδεσης
- Δυναμική σύνδεση (dynamic linking)
  - Τα αντικείμενα (object files) παραμένουν αποσυνδεδεμένα
  - Ο κώδικας αντιστοιχείται και συνδέεται κατά τον χρόνο εκτέλεσης. Παράδειγμα αποτελούν οι βιβλιοθήκες συστήματος (system libraries).



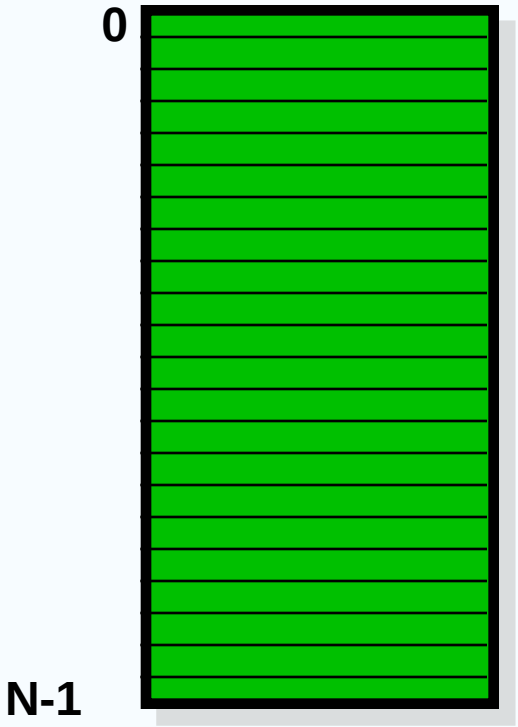
Ιδεατές και πραγματικές διευθύνσεις

- Τα συστήματα ιδεατής μνήμης καλύπτουν τις ανάγκες των διεργασιών μέσω της ψευδαίσθησης ότι έχουν στη διάθεσή τους περισσότερη κύρια μνήμη από όση διαθέτει το υπολογιστικό σύστημα.
- Έτσι υπάρχουν δύο τύποι διευθύνσεων στα συστήματα ιδεατής μνήμης :
  - Αυτές στις οποίες αναφέρονται οι διεργασίες:
    - **Ιδεατές ή εικονικές διευθύνσεις** – virtual addresses
  - Αυτές που είναι διαθέσιμες στην κύρια μνήμη:
    - **Φυσικές ή πραγματικές διευθύνσεις** – real addresses



Ιδεατή και φυσική μνήμη

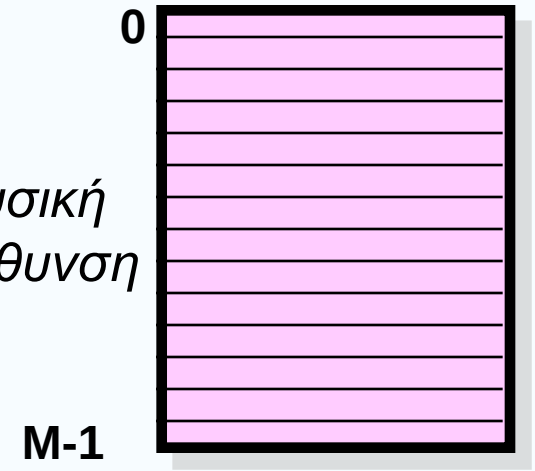
Ιδεατή μνήμη
(ονόματα δεδομένων)

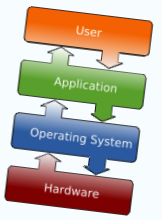


μετάφραση

Ιδεατή διεύθυνση → *Φυσική διεύθυνση*

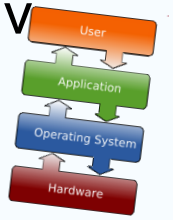
Φυσική μνήμη
(τοποθεσία δεδομένων)





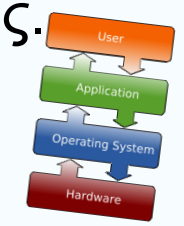
Λογικές, σχετικές & φυσικές διευθύνσεις

- Μια **λογική διεύθυνση** είναι μια αναφορά σε μια θέση μνήμης ανεξάρτητη από τη φυσική δομή και οργάνωση της μνήμης.
  - Οι compilers παράγουν κώδικα στον οποίο όλες οι αναφορές μνήμης είναι λογικές διευθύνσεις.
  - Οι λογικές διευθύνσεις παράγονται από τον επεξεργαστή.
  - Πριν πραγματοποιηθεί η πρόσβαση στην κύρια μνήμη γίνεται μετάφραση στη φυσική διεύθυνση
- Μια **σχετική διεύθυνση** είναι ένα παράδειγμα λογικής διεύθυνσης στο οποίο η διεύθυνση εκφράζεται ως μια θέση σχετική με ένα γνωστό σημείο του προγράμματος (π.χ. η αρχή του τμήματος μνήμης)
- Μια **φυσική ή απόλυτη διεύθυνση** είναι μια πραγματική θέση στην κύρια μνήμη

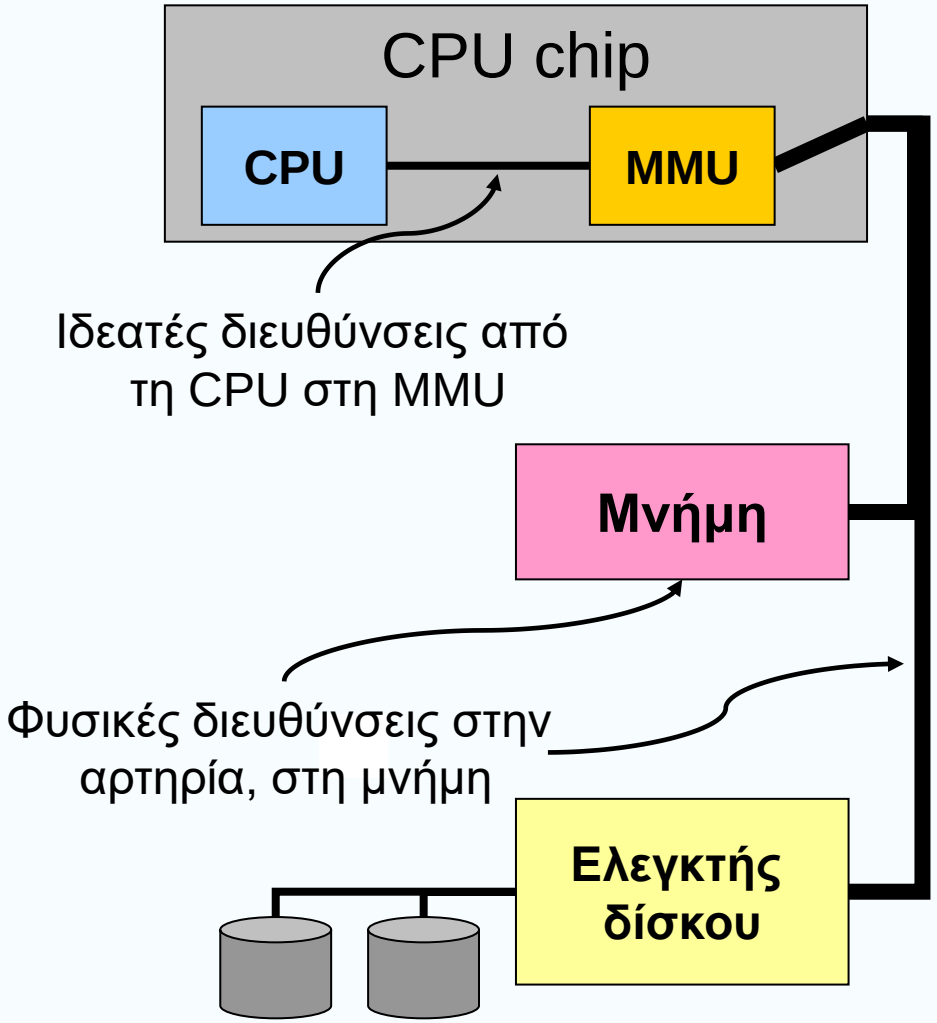


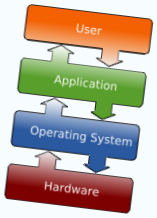
Μετάφραση διεύθυνσης

- Οι σχετικές διευθύνσεις είναι ο πλέον συνηθισμένος τύπος λογικών διευθύνσεων που χρησιμοποιούνται στα εκτελέσιμα αρχεία (modules προγραμματισμού).
- Οι φυσικές διευθύνσεις «υπολογίζονται» καθώς εκτελούνται οι εντολές.
- Για να υπάρχει επαρκής απόδοση, η μετάφραση από σχετικές σε φυσικές διευθύνσεις γίνεται από το υλικό.
- Τα προγράμματα των χρηστών διαχειρίζονται μόνον λογικές διευθύνσεις και δεν «βλέπουν» ποτέ τις πραγματικές φυσικές διευθύνσεις.
- Η μονάδα διαχείρισης μνήμης (**Memory Management Unit - MMU**) είναι μια συσκευή που αντιστοιχεί τις εικονικές σε φυσικές διευθύνσεις.

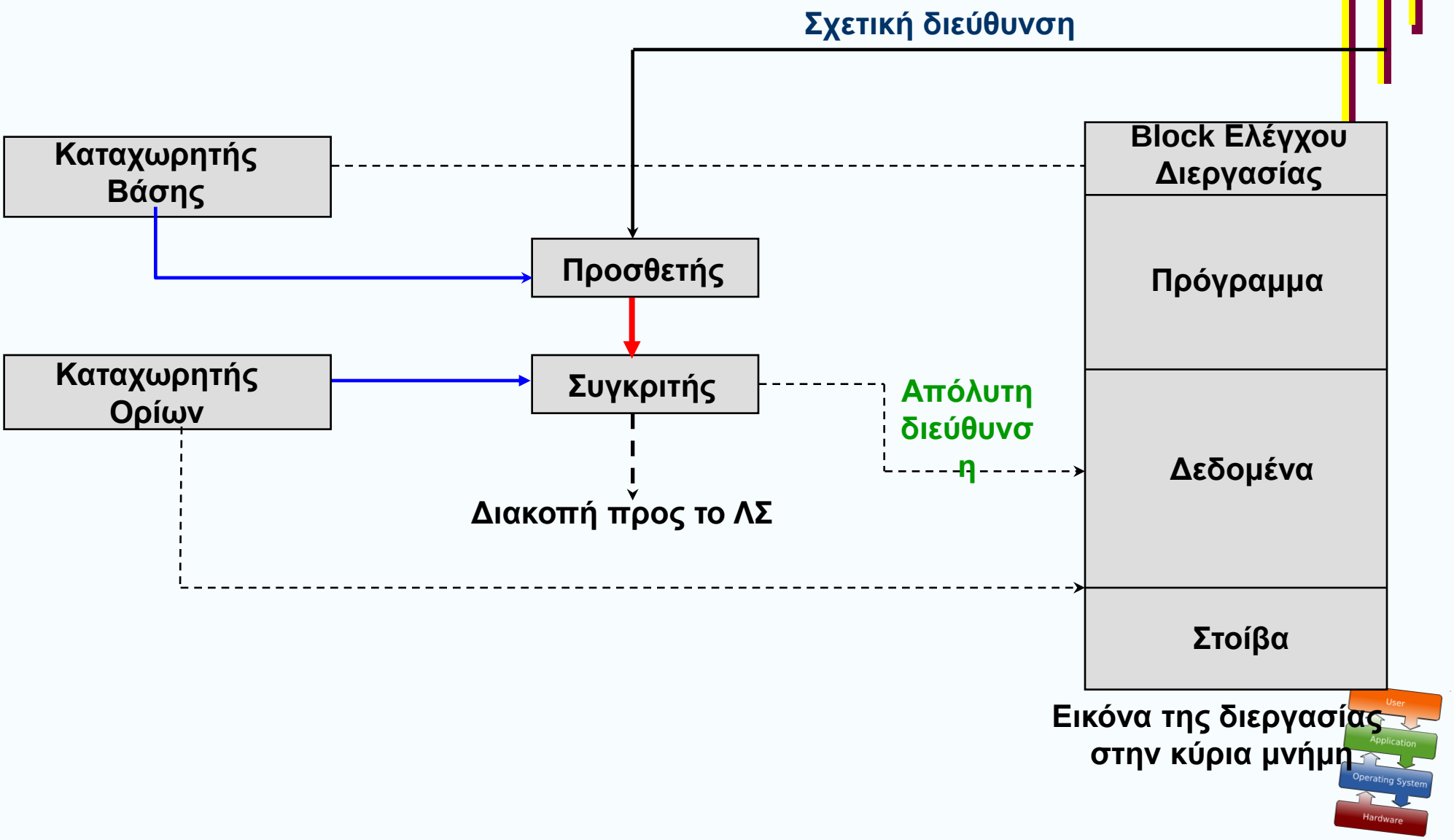


Η θέση και η λειτουργία της MMU

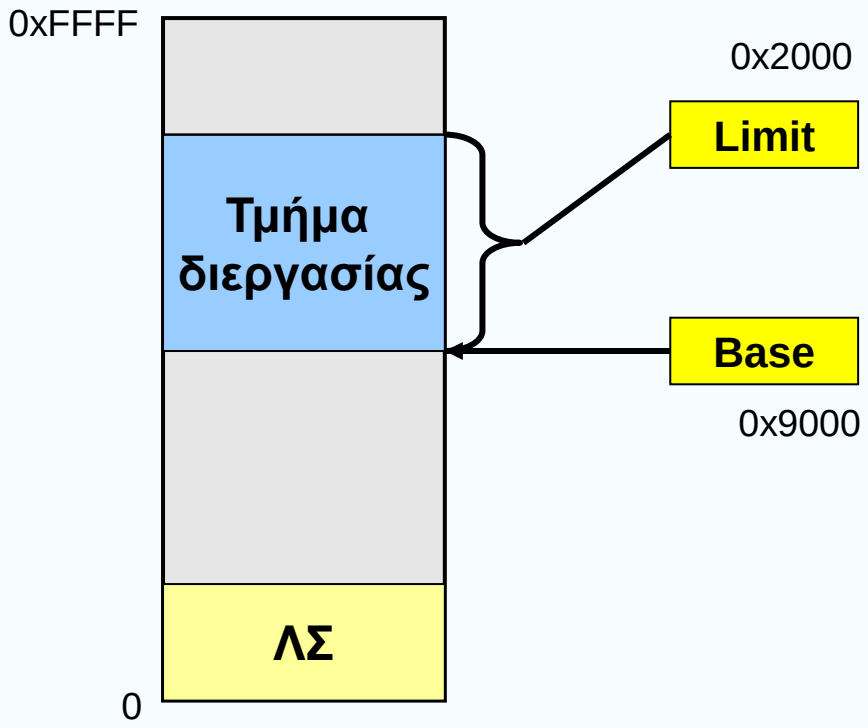




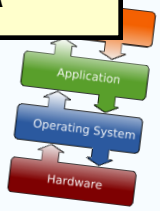
Υποστήριξη υλικού μέρους για τη μετατόπιση



Καταχωρητές βάσης και ορίου

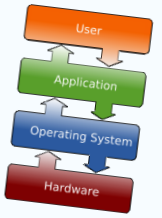


Λογική διεύθυνση: **0x1204**
Φυσική διεύθυνση: **0x1204**+**0x9000** = **0xa204** < **0x2000** → δεν υπάρχει σφάλμα



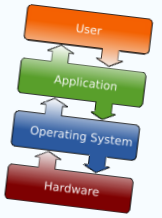
Λογική οργάνωση μνήμης

- Η κύρια μνήμη σε ένα υπολογιστικό σύστημα οργανώνεται ως ένας γραμμικός, μονοδιάστατος χώρος διευθύνσεων
- Η δευτερεύουσα μνήμη, σε φυσικό επίπεδο, οργανώνεται με παρόμοιο τρόπο.
- Τα προγράμματα οργανώνονται και γράφονται σε ενότητες (modules).
- Οι ενότητες αυτές γράφονται και μεταφράζονται ανεξάρτητα.
- Στις ενότητες δίνονται διαφορετικοί βαθμοί προστασίας (read-only, execute-only)
- Οι ενότητες μπορούν να διαμοιράζονται μεταξύ των διεργασιών



Μνήμη και πολυπρογραμματισμός

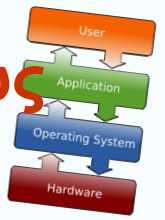
- Ο πολυπρογραμματισμός είναι απαραίτητος για την αποτελεσματικότητα των συστημάτων.
- Η μνήμη για τον πολυπρογραμματισμό χρειάζεται
  - Επανατοποθέτηση (Relocation)
  - Προστασία (Protection)
- Το ΛΣ δεν μπορεί να είναι βέβαιο σε ποιο τμήμα της μνήμης θα φορτωθεί ένα πρόγραμμα
  - Μεταβλητές και διαδικασίες δεν μπορούν να χρησιμοποιούν απόλυτες διευθύνσεις μνήμης



Μνήμη και πολυπρογραμματισμός (συν.)

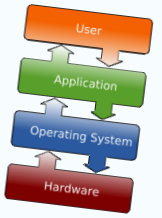
- Το ΛΣ πρέπει να διατηρεί ξεχωριστά τη μνήμη για κάθε διεργασία
  - Προστασία μιας διεργασίας από άλλες που θέλουν να διαβάσουν ή να γράψουν στη δική της περιοχή μνήμης
  - Προστασία μιας διεργασίας από την τροποποίηση της δικής της μνήμης με ανεπιθύμητο τρόπο (πχ γράφοντας στο τμήμα κώδικα)
- Το ΛΣ πρέπει να επιτρέπει σε πολλές διεργασίες να έχουν πρόσβαση στην ίδια περιοχή της μνήμης.
  - Είναι προτιμότερο να επιτρέπεται η πρόσβαση σε μια διεργασία (σε ένα άτομο) στο ίδιο αντίγραφο του προγράμματος από το να υπάρχει ένα αντίγραφο για κάθε μια διεργασία.

Για να ικανοποιηθούν αυτές τις ανάγκες χρησιμοποιείται ως εργαλείο η τμηματοποίηση της ιδεατής μνήμης.



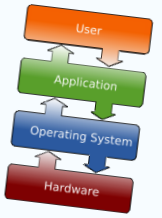
Τμηματοποίηση ιδεατής μνήμης

- Η ιδεατή μνήμη χρησιμοποιεί τρεις βασικές τεχνικές :
  - Σελιδοποίηση (paging)
  - Κατάτμηση (segmentation)
  - Κατάτμηση με σελιδοποίηση (segmentation with paging)

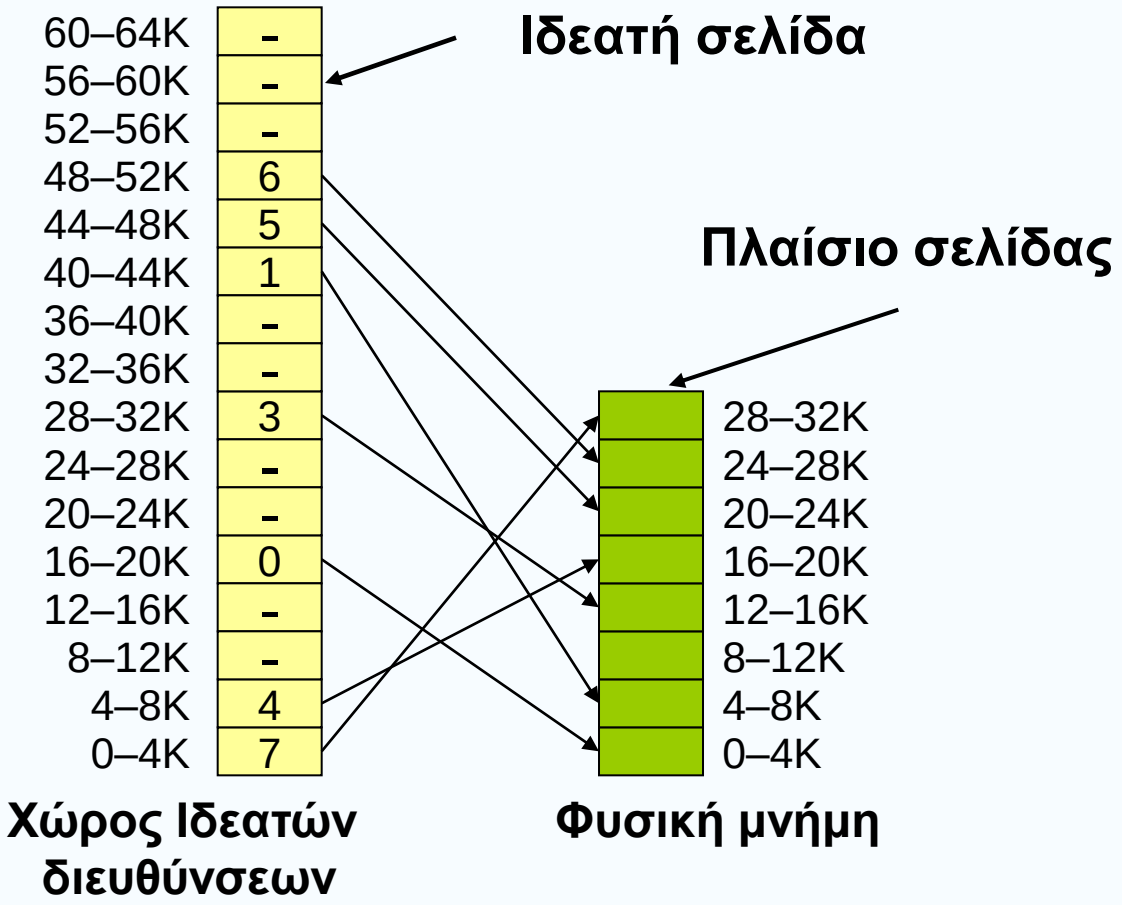


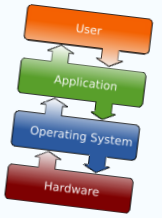
Σελιδοποίηση

- Η κατάτμηση της μνήμης σε μικρά ίσου μεγέθους τμήματα (blocks, chunks) και η διαίρεση κάθε διεργασίας σε τμήματα του ίδιου μεγέθους
- Τα τμήματα μιας διεργασίας λέγονται **σελίδες (pages)** και τα τμήματα της μνήμης **πλαίσια (frames).**
- Ο εικονικός χώρος διευθύνσεων διαμοιράζεται σε σελίδες σταθερού μεγέθους, ενώ η φυσική μνήμη διαμοιράζεται σε πλαίσια σελίδας (page frames**)** (μεγέθους ίδιου με τη σελίδα).
- Μια σελίδα μπορεί να τοποθετηθεί σε οποιοδήποτε πλαίσιο σελίδας



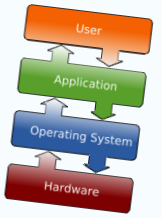
Σελίδες και πλαίσια



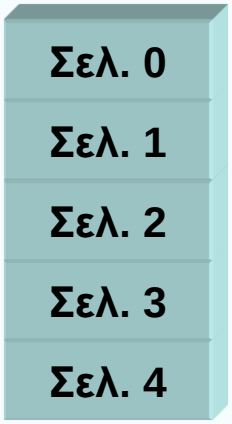


Σελίδες και πλαίσια (συν.)

- Το ΛΣ διατηρεί
  - ένα πίνακα σελίδων (**page table**) για κάθε διεργασία που περιέχει τη θέση πλαισίου για κάθε σελίδα της διεργασίας.
  - μια λίστα των ελεύθερων πλαισίων, με όλα τα πλαίσια στην κύρια μνήμη που είναι ελεύθερα και διαθέσιμα για τις σελίδες.
- Η διεύθυνση μνήμης μέσα στο πρόγραμμα (λογική διεύθυνση) αποτελείται από έναν **αριθμό σελίδας** και μια **μετατόπιση** (offset) εντός της σελίδας
- Ο επεξεργαστής χρησιμοποιεί τον πίνακα σελίδων για να παράγει τη **φυσική διεύθυνση (αριθμός πλαισίου, μετατόπιση)** που αντιστοιχεί σε μια **λογική διεύθυνση (αριθμός σελίδας, μετατόπιση)**

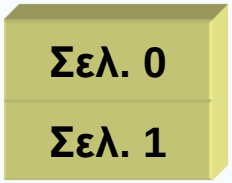


Αντιστοίχηση σελίδων διεργασιών με ελεύθερα πλαίσια

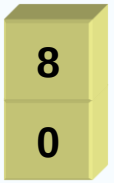


Λογική μνήμη (P0)

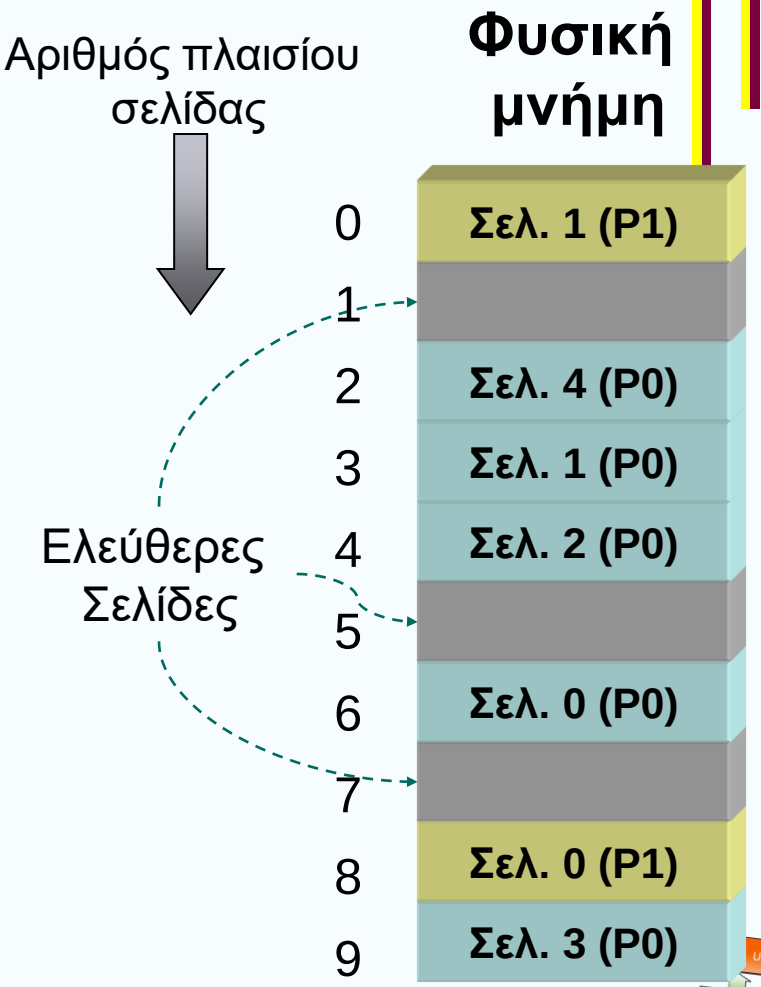
Πίνακας Σελίδων (P0)

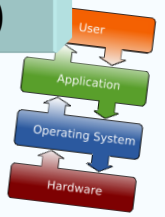


Λογική Μνήμη (P1)



Πίνακας Σελίδων (P1)



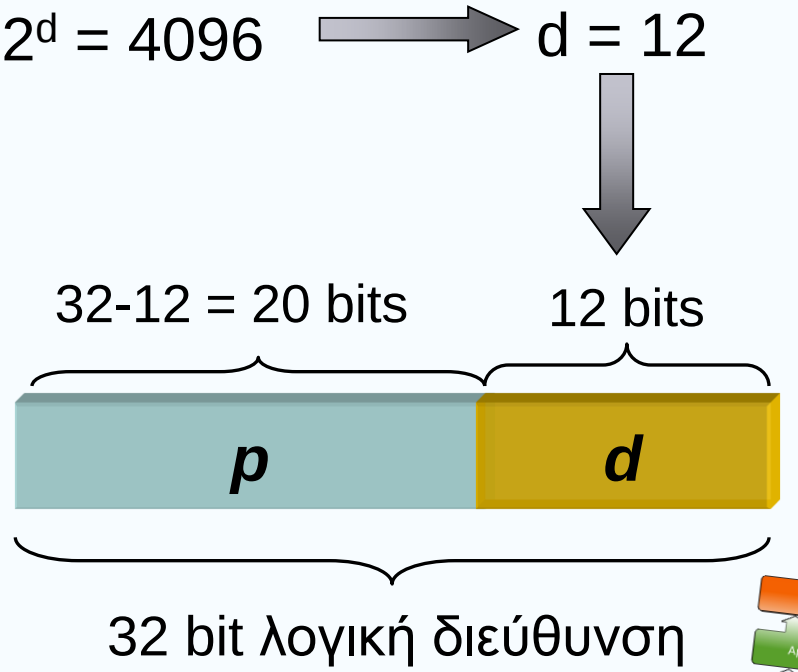


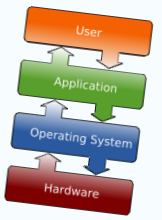
Αντιστοίχηση λογικών σε φυσικές διευθύνσεις

- Διαίρεση της διεύθυνσης από τη CPU σε δύο τμήματα
  - Αριθμός σελίδας (*p*)
  - Μετατόπιση στη σελίδα (*d*)
- Αριθμός σελίδας
  - Δείκτης στον πίνακα σελίδων
  - Ο πίνακας σελίδων περιέχει τη διεύθυνση βάσης της σελίδας στη φυσική μνήμη
- Μετατόπιση στη σελίδα
  - Προστίθεται στη διεύθυνση βάσης για να βρεθεί η πραγματική διεύθυνση στη φυσική μνήμη
- Μέγεθος σελίδας = 2^d bytes

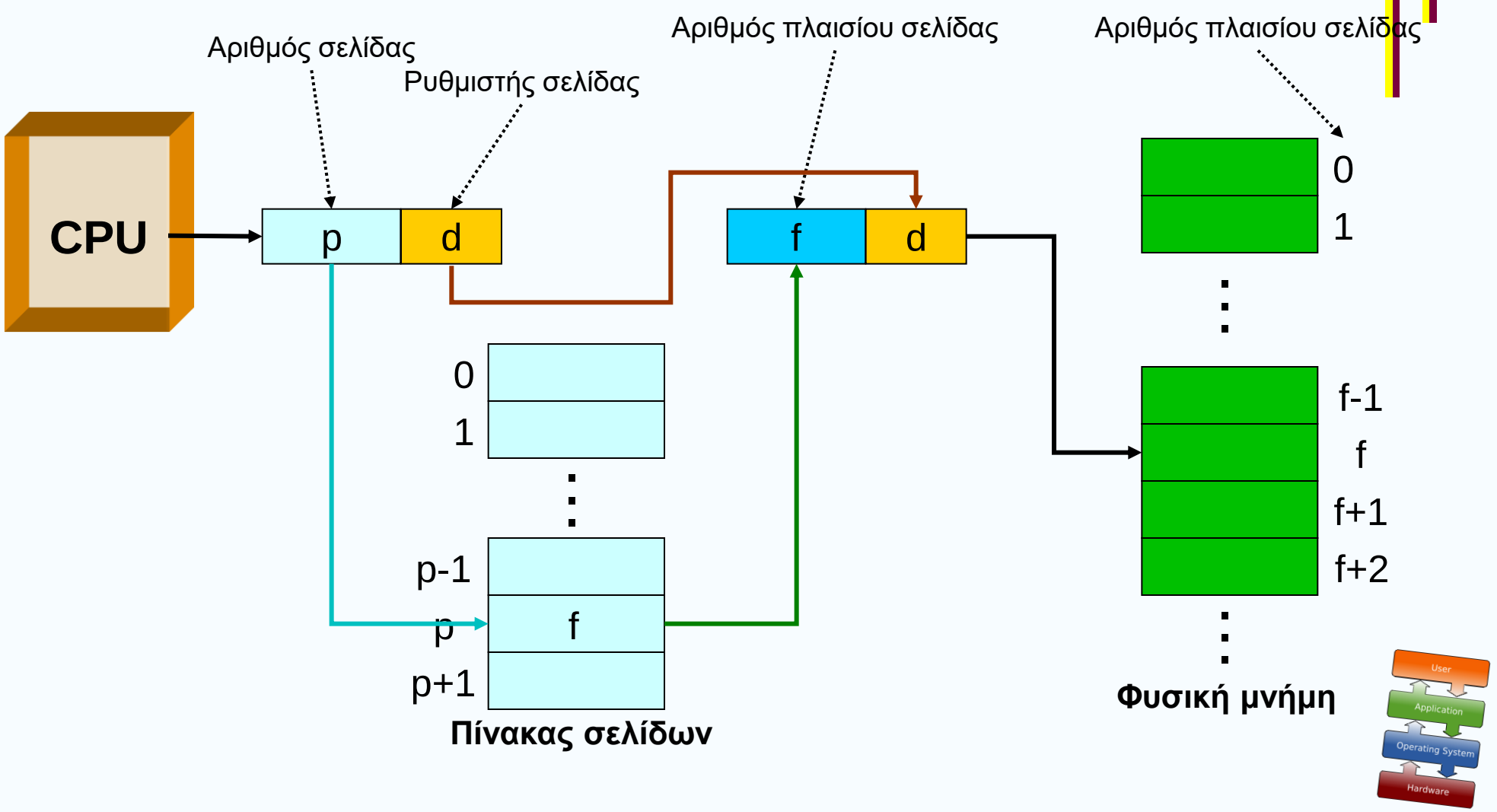
Παράδειγμα:

- 4 KB (=4096 byte) σελίδες
- 32 bit λογικές διευθύνσεις





Αρχιτεκτονική μετάφρασης της διεύθυνσης



Παράδειγμα

Πίνακας σελίδων

		Παρόν bit
15	000	0
14	000	0
13	000	0
12	000	0
11	111	1
10	000	0
9	101	1
8	000	0
7	000	0
6	000	0
5	011	1
4	100	1
3	000	1
2	110	1
1	001	1
0	010	1

Τελική φυσική διεύθυνση

(0x6004, 24580)

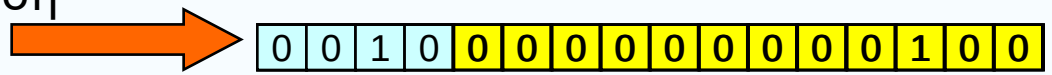
Παράδειγμα:

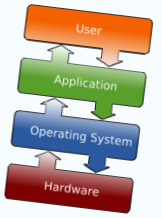
- **4 KB σελίδες (12-bit offsets)**
- 16 bit χώρος ιδεατών διευθύνσεων → 16 σελίδες (4-bit index)
- 8 φυσικές σελίδες (3-bit index)

4-bit δείκτης
Στον πίνακα σελίδων
virtual page = 0010 = 2

12-bit offset

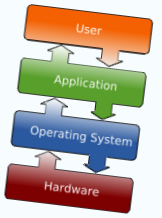
Εισερχόμενη Ιδεατή διεύθυνση
(0x2004, 8196)





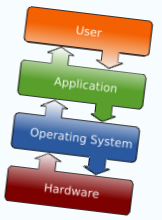
Σελιδοποίηση

- Η σελιδοποίηση είναι ανάλογη με την τμηματοποίηση σταθερού μεγέθους, με τις εξής διαφορές:
  1. τα τμήματα δεν χρειάζεται να είναι συνεχόμενα
  2. τα τμήματα είναι αρκετά μικρά
  3. ένα πρόγραμμα μπορεί να απασχολεί περισσότερα από ένα τμήματα
- Η σπατάλη μνήμης οφείλεται στον εσωτερικό κατακερματισμό που είναι κλάσμα της τελευταίας σελίδας της διεργασίας. Εξωτερικός κατακερματισμός δεν υπάρχει.



Σελιδοποίηση (συν.)

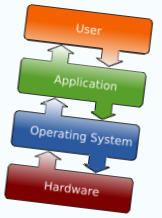
- **Το μέγεθος σελίδας και πλαισίου είναι δύναμη του 2** (συνήθως μεταξύ 512 bytes και 8192 bytes).
- Αποδεικνύεται ότι η σχετική διεύθυνση που ορίζεται σε σχέση με την αρχή του προγράμματος και η λογική διεύθυνση που εκφράζεται ως ένας αριθμός σελίδας και μετατόπιση, είναι **ΙΔΙΕΣ**.
- Οφέλη της χρήσης μεγεθών σελίδας που είναι δυνάμεις του 2:
  - Το σχήμα της λογικής διευθυνσιοδότησης δεν είναι εμφανές στον προγραμματιστή.
  - Είναι εύκολη η εφαρμογή μιας συνάρτησης σε επίπεδο υλικού μέρους που θα πραγματοποιεί δυναμική μετάφραση των διευθύνσεων κατά τη στιγμή της εκτέλεσης.



Μέγεθος σελίδας

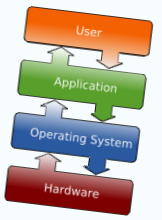
Μικρό μέγεθος σελίδας

- Πλεονεκτήματα
  - λιγότερος εσωτερικός κατακερματισμός
  - καλύτερο ταίριασμα για διάφορες δομές δεδομένων και τμήματα κώδικα
  - λιγότερο μη χρησιμοποιούμενο πρόγραμμα στη μνήμη
- Μειονεκτήματα
  - τα προγράμματα χρειάζονται πολλές σελίδες και μεγαλύτερους πίνακες σελίδων

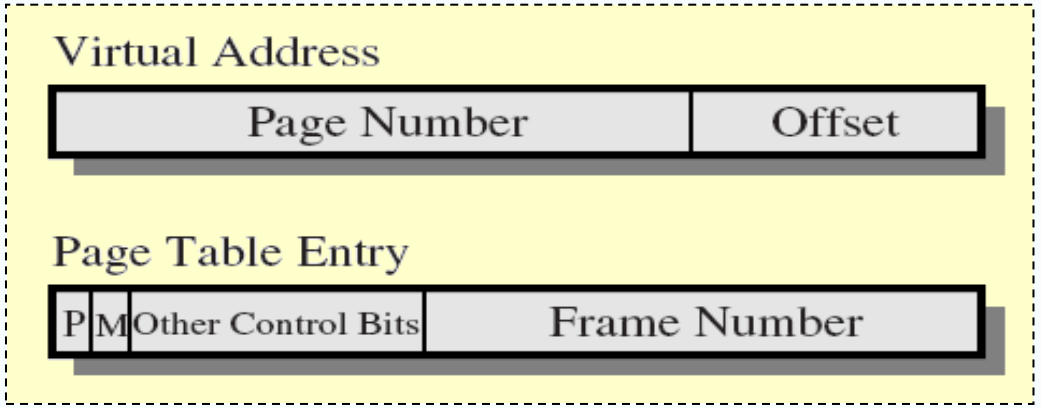


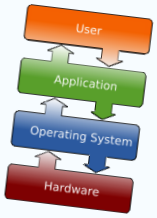
Πίνακες σελίδων

- Κάθε διεργασία έχει τον δικό της εικονικό χώρο διευθύνσεων, άρα και τον δικό της πίνακα σελίδων που αποθηκεύεται στην κύρια μνήμη.
- **Κάθε καταχώρηση (θέση) του πίνακα σελίδων περιέχει τον αριθμό πλαισίου της αντίστοιχης σελίδας στην κεντρική μνήμη**
- Επιπλέον απαιτούνται :
  - ένα bit (**P**resent bit) που θα δείχνει αν μια σελίδα βρίσκεται στην κεντρική μνήμη ή όχι
  - ένα bit (**M**odify bit) για να δείχνει αν η σελίδα έχει αλλάξει από την τελευταία φορά που φορτώθηκε στην κεντρική μνήμη. Αν δεν έχει γίνει αλλαγή, τότε η σελίδα δεν πρέπει να γραφεί στο δίσκο όταν χρειάζεται να γίνει εναλλαγή.

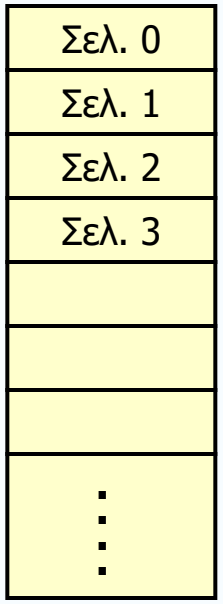


Είσοδοι πίνακα σελίδων

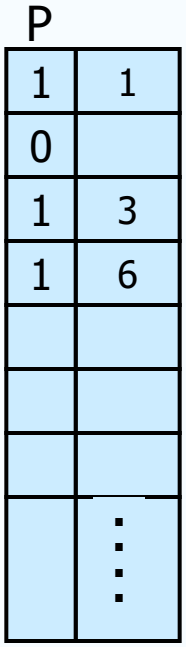




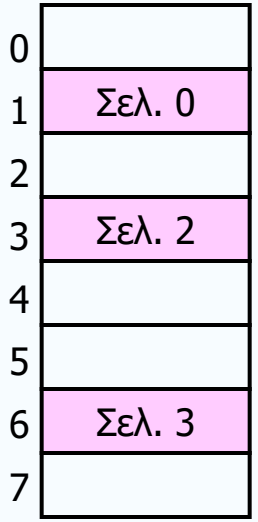
Παράδειγμα



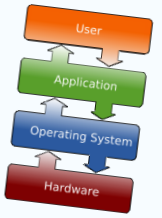
Ιδεατή Μνήμη



Πίνακας Σελίδων



Φυσική Μνήμη



Πολλαπλές διεργασίες στη φυσική μνήμη

Διεργασία Α
Ιδεατή μνήμη

Σελ. Α0
Σελ. Α1
Σελ. Α2
Σελ. Α3
⋮

Πίνακας Σελίδων Α

1	1
0	
1	3
1	6
	⋮

Διεργασία Β
Ιδεατή μνήμη

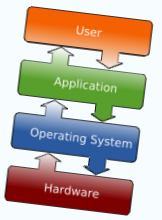
Σελ. Β0
Σελ. Β1
Σελ. Β2
Σελ. Β3
⋮

Πίνακας σελίδων Β

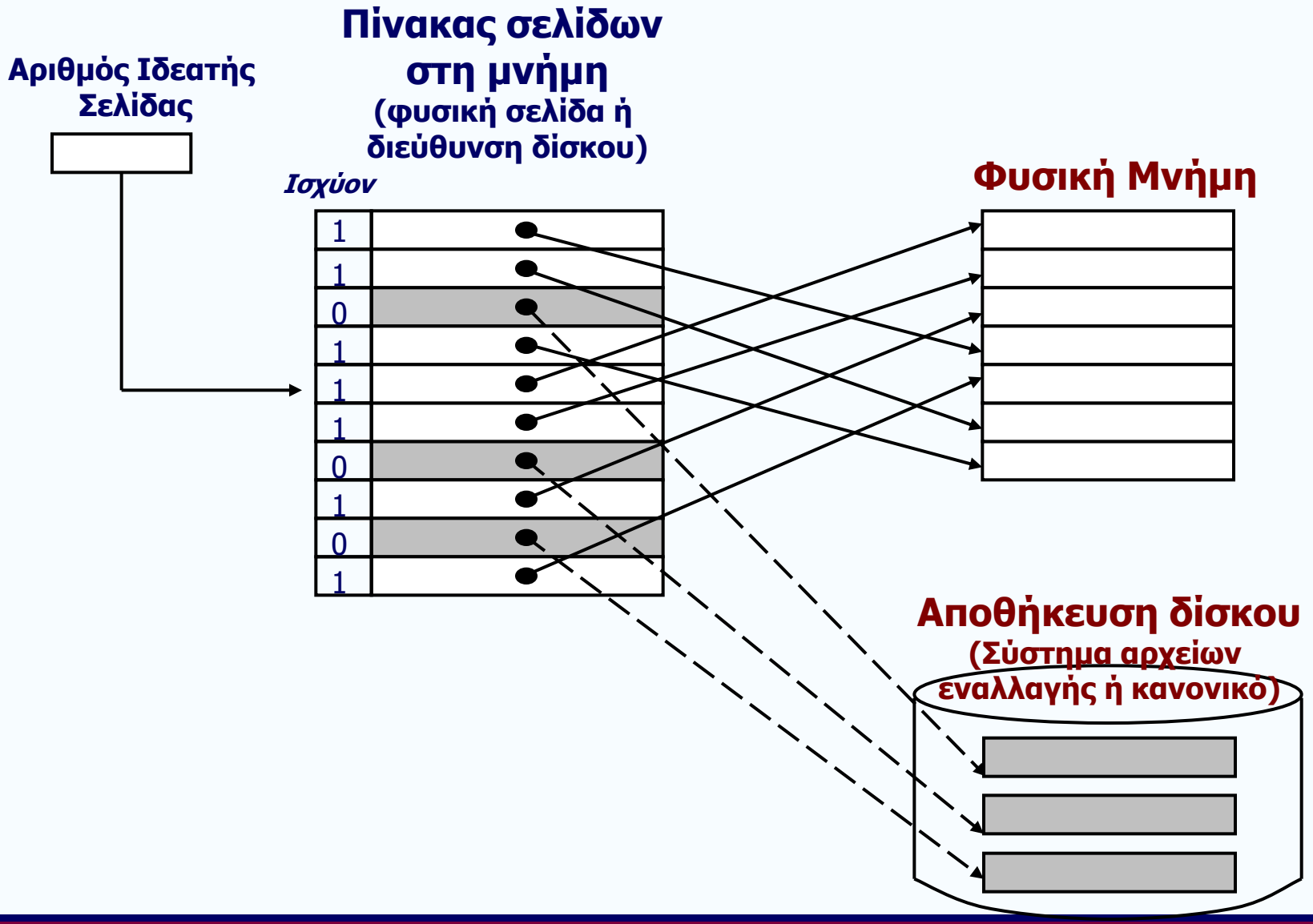
1	8
1	4
0	
1	10
	⋮

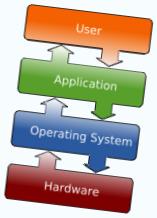
Φυσική μνήμη

0	
1	Σελ. Α0
2	
3	Σελ. Α2
4	Σελ. Β1
5	
6	Σελ. Α3
7	
8	Σελ. Β0
9	
10	Σελ. Β3
11	



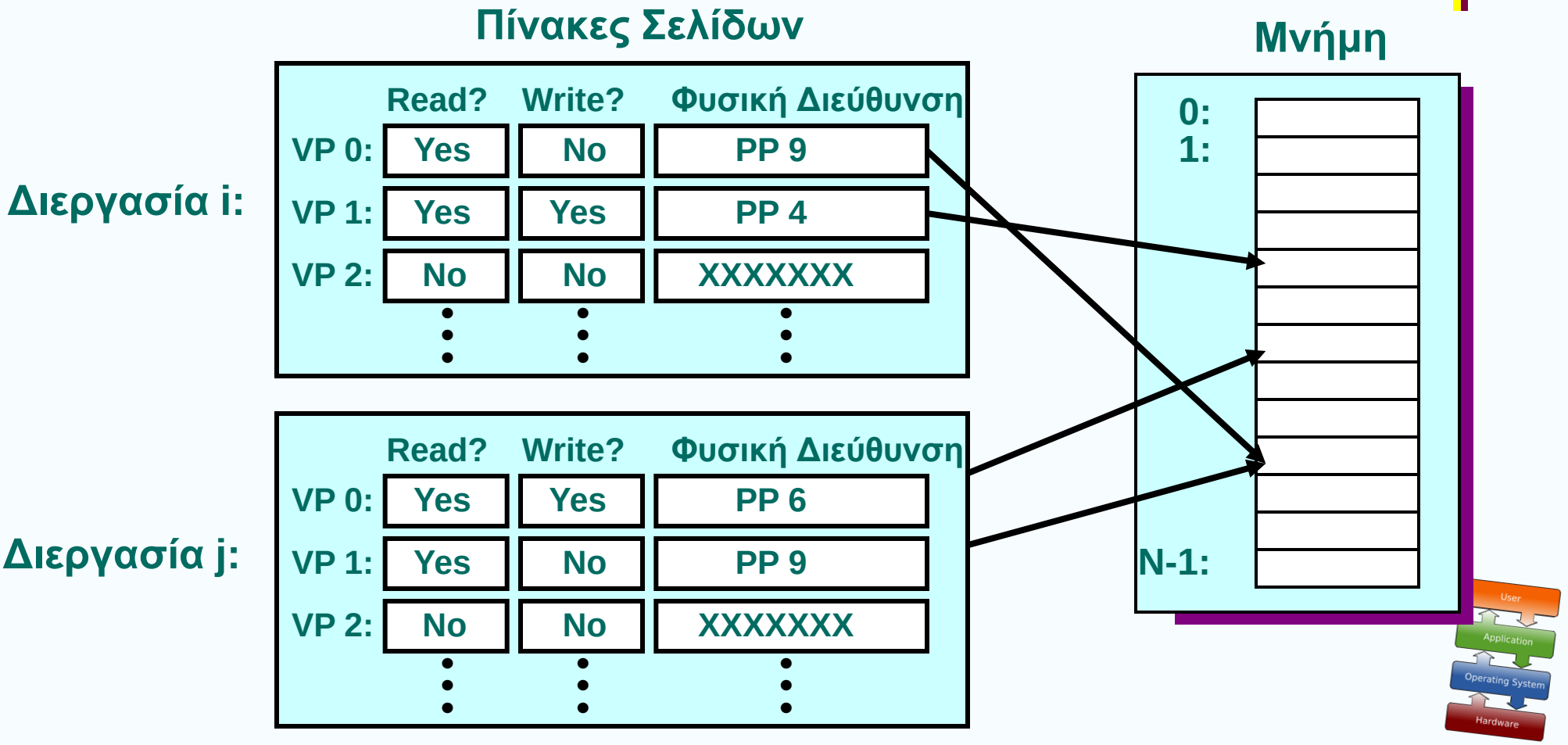
Πίνακες σελίδων – Ισχύον bit





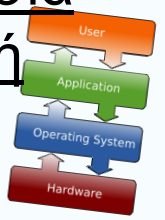
Προστασία

- Κάθε είσοδος στον πίνακα σελίδων περιέχει πληροφορίες για τα δικαιώματα πρόσβασης (access rights)



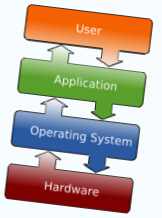
Πίνακες σελίδων - συνολικά

- Ο πίνακας σελίδων έχει μέγιστο πλήθος εισόδων όσο και το πλήθος των εικονικών σελίδων της διεργασίας.
- Ο πίνακας σελίδων είναι μεταβλητού μήκους (το μήκος του εξαρτάται από το μέγεθος της διεργασίας) και πρέπει να βρίσκεται στην κύρια μνήμη για να είναι προσπελάσιμος.
- Ολόκληρος ο πίνακας σελίδων είναι πιθανόν να καταλαμβάνει πολύ μεγάλο μέρος της κεντρικής μνήμης.
- **Το μέγεθος της μνήμης που αφιερώνεται στους πίνακες σελίδων μπορεί να γίνει απαράδεκτα μεγάλο.**
  - Μεγαλύτερο μέγεθος σελίδας είναι μια ενδεδειγμένη λύση;
- Τα περισσότερα συστήματα ιδεατής μνήμης αποθηκεύουν τους πίνακες των σελίδων στην ιδεατή μνήμη, αντί για τη φυσική. Όταν μια διεργασία εκτελείται, ένα τμήμα του πίνακα σελίδων της βρίσκεται στην κεντρική μνήμη.

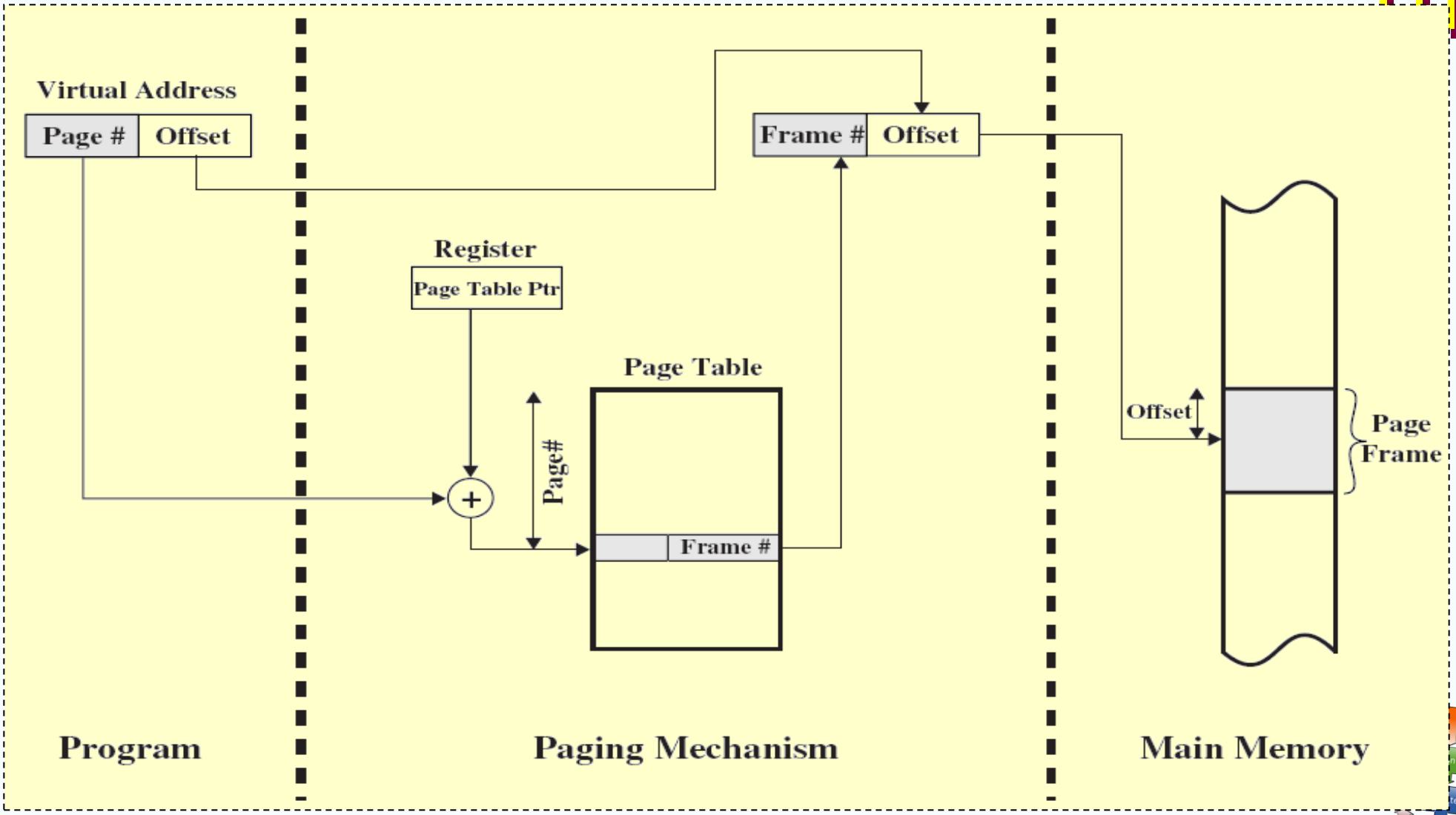


Πίνακες σελίδων – συνολικά (συν.)

- Κάθε πίνακας σελίδων χρησιμοποιεί τους καταχωρητές :
  - Page Table Base Register (PTBR):
    - κρατά την αρχική φυσική διεύθυνση του πίνακα σελίδων της εκτελούμενης διεργασίας
  - Page-table length register (PRLR):
    - το υλικό ελέγχει αν ο αριθμός σελίδας βρίσκεται εντός του ορίου και το μέγεθος του πίνακα σελίδων περιορίζεται.
- Ορισμένοι επεξεργαστές χρησιμοποιούν πίνακες σελίδων πολλαπλών επιπέδων για να οργανώσουν καλύτερα μεγάλους πίνακες σελίδων.



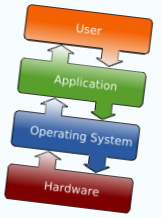
Μετάφραση διεύθυνσης σε σύστημα σελιδοποίησης



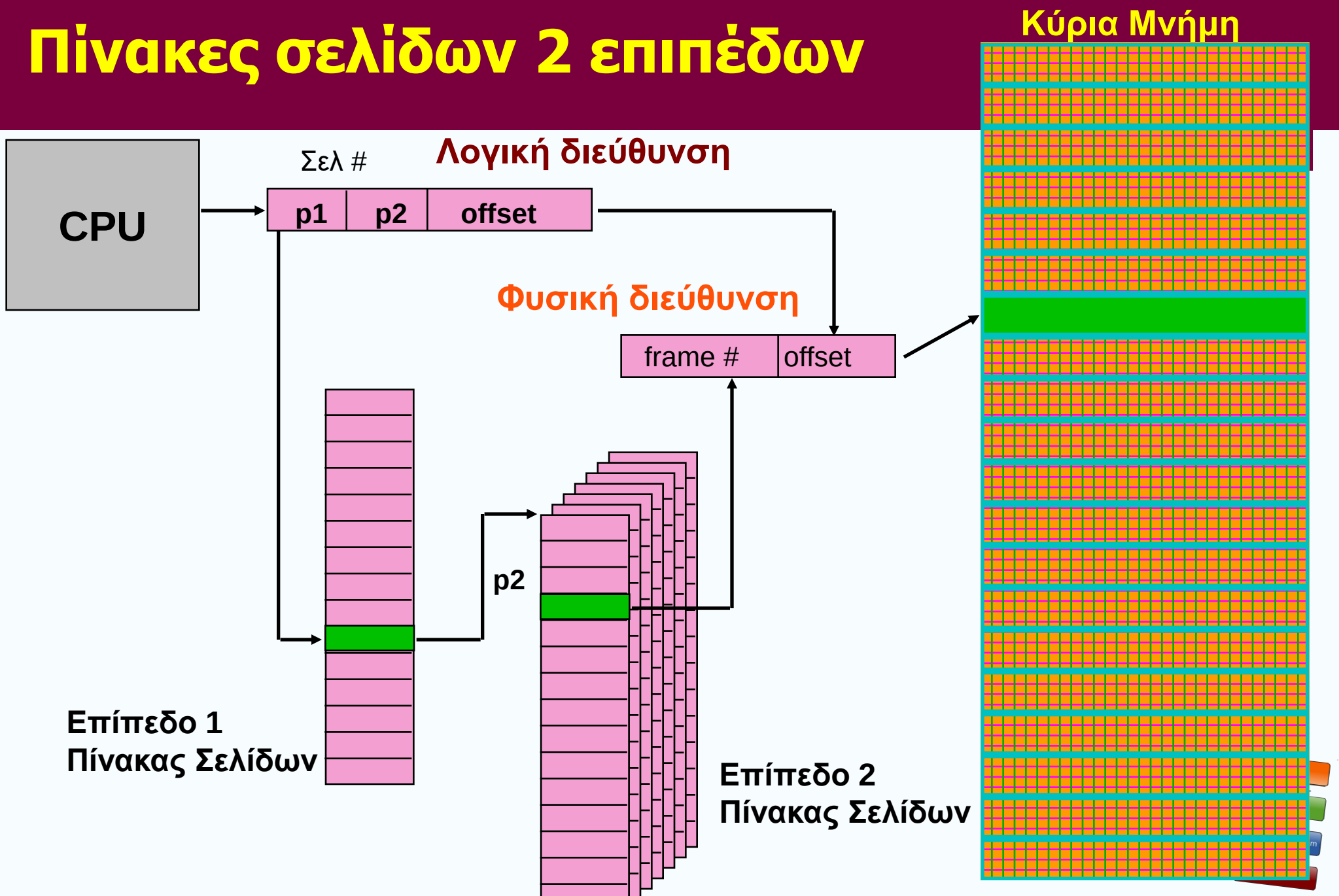
Παράδειγμα

- Αν χρησιμοποιούνται 32 bits για εικονικό χώρο διευθύνσεων με μέγεθος σελίδων 4KB τότε ένας πίνακας σελίδων μπορεί να έχει 2^{20} εισόδους (θέσεις).
- Αν κάθε θέση του καταλαμβάνει 4 bytes τότε δεσμεύεται χώρος κύριας μνήμης :

  2^{20} X 4 bytes = 2^{22} bytes = 4 MB
- Αν στο σύστημα «τρέχουν» 25 διεργασίες τότε απαιτείται χώρος για τους πίνακες σελίδων : 25 X 4 MB = 100 MB!!!!
- Μέγεθος φυσικής μνήμης 4Gb είναι συνηθισμένο και πιθανό;
- Ήδη χρησιμοποιείται διευθυνσιοδότηση 64-bits για την αύξηση του εικονικού χώρου διευθύνσεων:
  - **Intel Itanium, AMD Clawhammer, DEC Alpha**

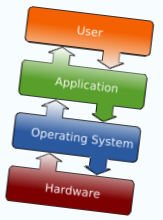


Πίνακες σελίδων 2 επιπέδων



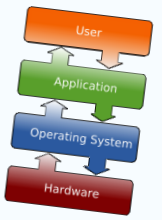
Translation Look-aside Buffer (TLB)

- Κάθε αναφορά στην ιδεατή μνήμη προκαλεί δύο προσβάσεις στη φυσική μνήμη.
  - Μία για την προσκόμιση (fetch) του πίνακα σελίδων, ώστε τελικά να δημιουργηθεί η φυσική διεύθυνση
  - Μία για την προσκόμιση των δεδομένων που υπάρχουν στη διεύθυνση
- Ένα άμεσο σχήμα ιδεατής μνήμης θα οδηγούσε σε διπλασιασμό του χρόνου προσπέλασης της μνήμης.
- Η βελτίωση της ταχύτητας πρόσβασης επιτυγχάνεται με τη χρήση ειδικής cache μνήμης υψηλής ταχύτητας που ονομάζεται :
  - **Translation Look-aside Buffer (TLB)** – ενδιάμεσος χώρος για τη μετάφραση σελίδων μνήμης που :
    - αποθηκεύει τις καταχωρήσεις του πίνακα σελίδων,
    - περιέχει τις καταχωρήσεις του πίνακα σελίδων που έχουν χρησιμοποιηθεί πρόσφατα και
    - λειτουργεί με τρόπο παρόμοιο με αυτόν της μνήμης cache
- **Το TLB αναφέρεται συχνά και με τον όρο** associative registers

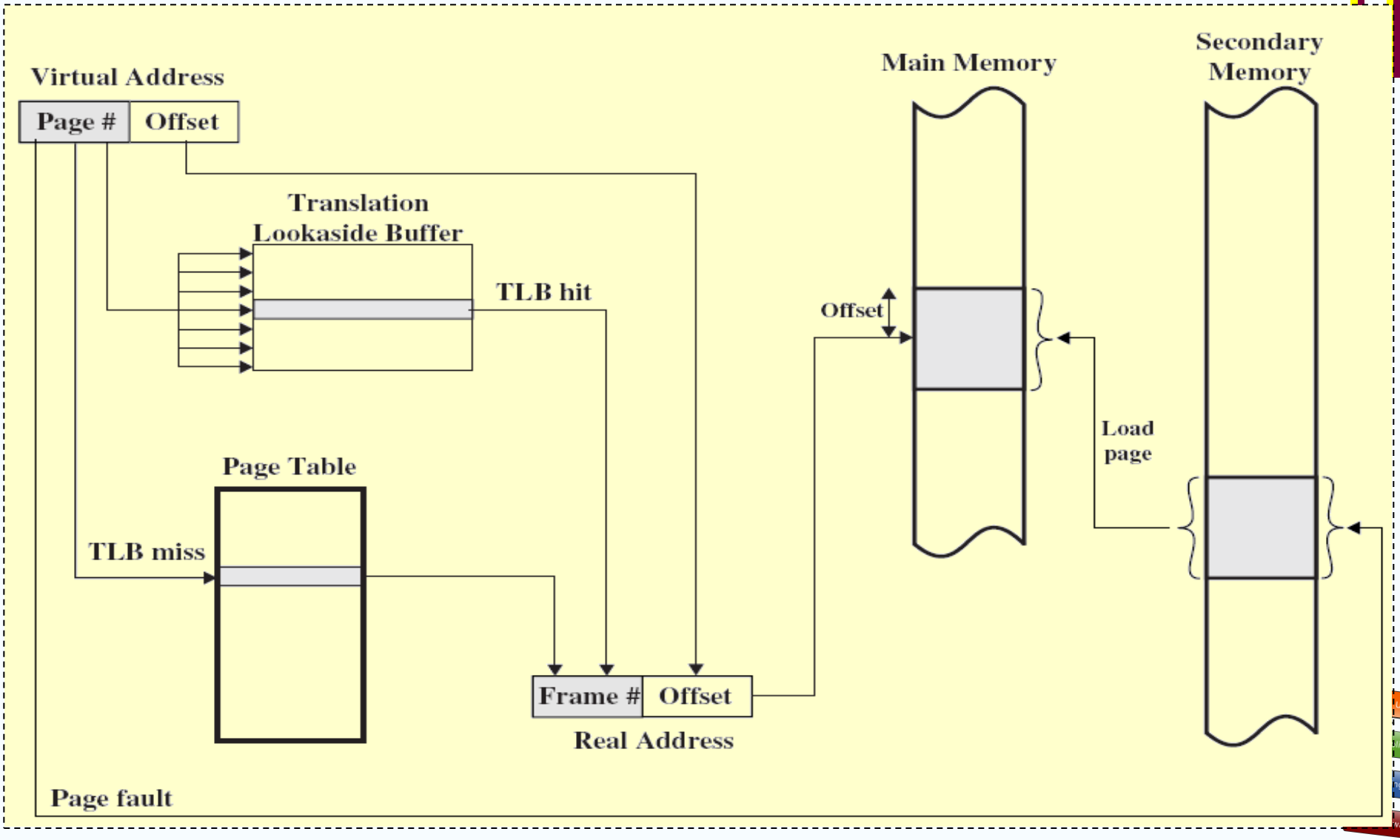


TLB (2)

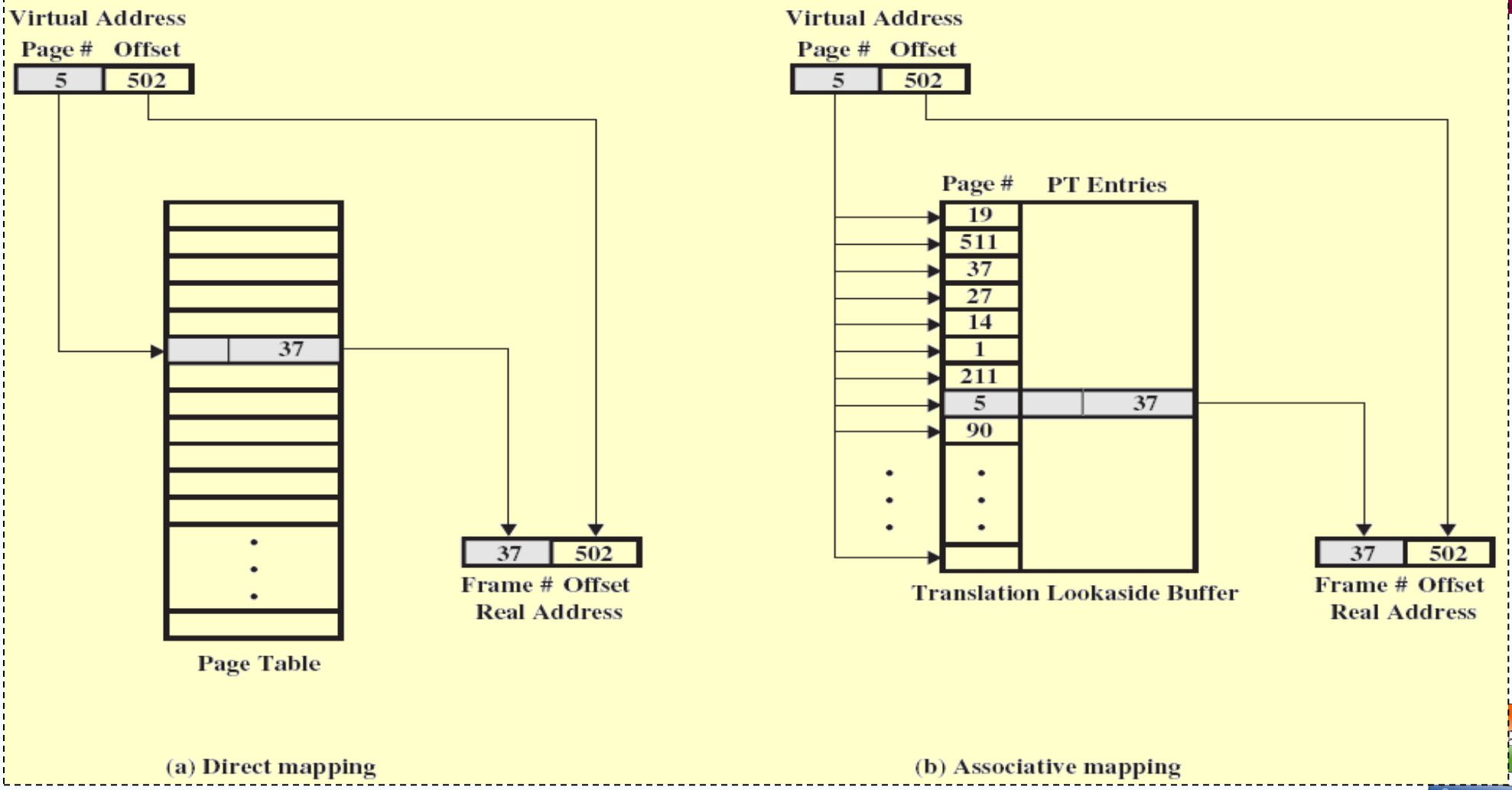
- Δεδομένης μιας εικονικής διεύθυνσης ο επεξεργαστής εξετάζει το TLB
- Αν η καταχώρηση στον πίνακα σελίδων υπάρχει (ευστοχία - hit) γίνεται ανάκτηση του αριθμού πλαισίου και δημιουργείται η πραγματική διεύθυνση.
- Αν η καταχώρηση στον πίνακα σελίδων δεν βρεθεί (αστοχία - miss), ο αριθμός σελίδας χρησιμοποιείται ως δείκτης στον πίνακα σελίδων της διεργασίας. Αν η ζητούμενη σελίδα υπάρχει εκεί ανακτάται και το TLB ενημερώνεται ώστε να συμπεριλάβει τη νέα είσοδο σελίδας. Αν η σελίδα δεν υπάρχει στον πίνακα σελίδων (άρα και στην κύρια μνήμη) απορρέει ένα σφάλμα σελίδας (page fault).
- Ως αναλογία επιτυχίας (hit ratio) ορίζεται το ποσοστό των επιτυχών αναζητήσεων μιας σελίδας στο TLB



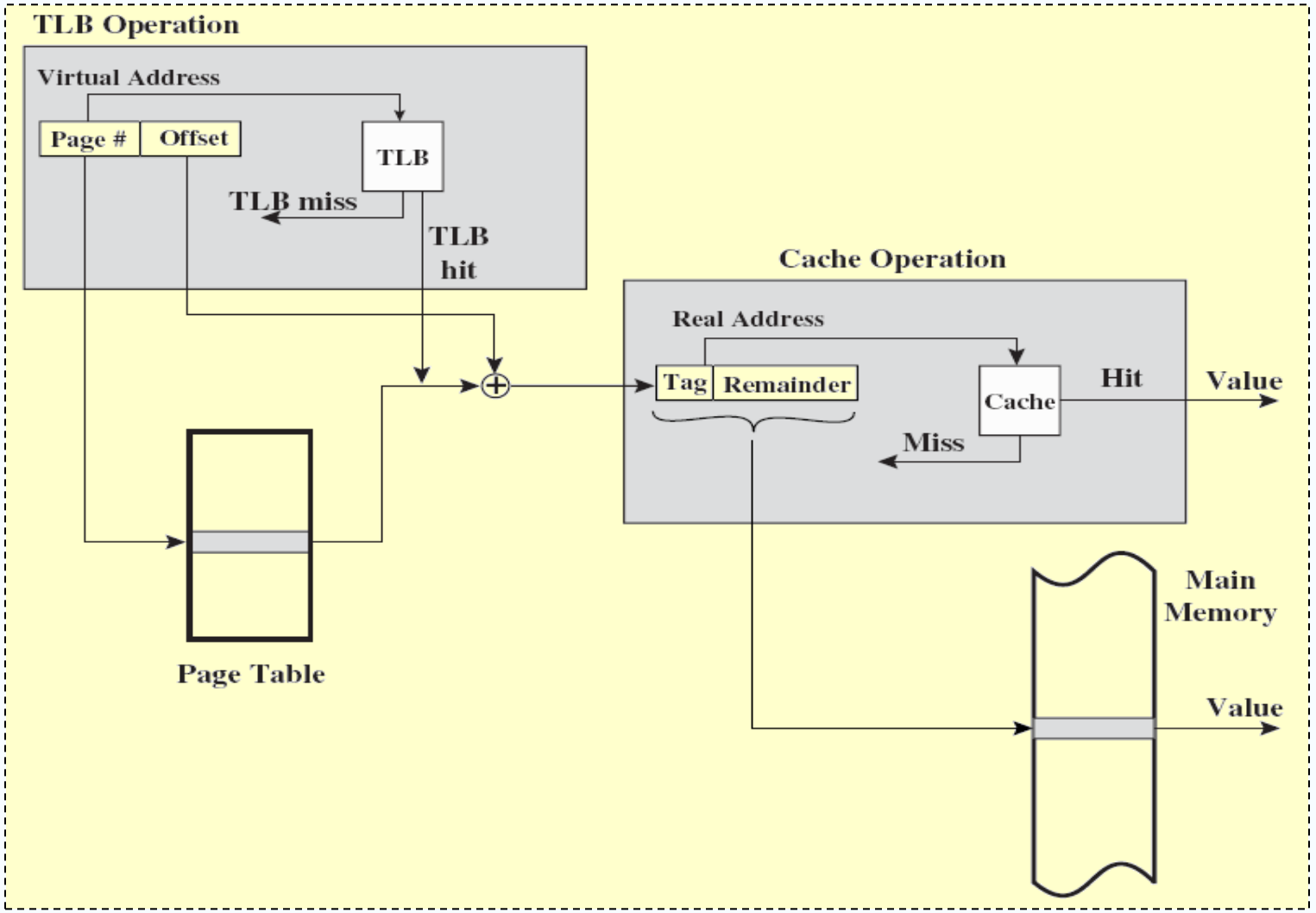
Μετάφραση διεύθυνσης σε σύστημα με TLB

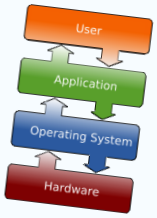


Σύγκριση άμεσης και συσχετιζόμενης αναζήτησης στον πίνακα σελίδων



Λειτουργία του TLB και κρυφή μνήμη

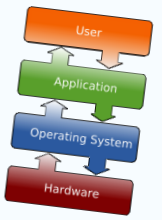




Αποδοτικός Χρόνος Πρόσβασης (Effective Access Time)

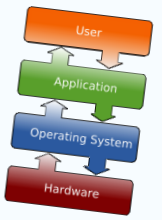
- Ο μέσος χρόνος προσπέλασης της κύριας μνήμης εξαρτάται από το πόσο συχνά ο ζητούμενος αριθμός σελίδας βρίσκεται στο TLB
  - Αυτό είναι γνωστό και ως *hit ratio*
- Ο πραγματικός χρόνος προσπέλασης (EAT) υπολογίζεται λαμβάνοντας υπόψη την περίπτωση να βρεθεί ή να μη βρεθεί η ζητούμενη σελίδα σύμφωνα με την πιθανότητα που δίνει το *hit ratio*

EAT = *hit ratioentryFound + (1-*hit ratio*)*entryNotFound**

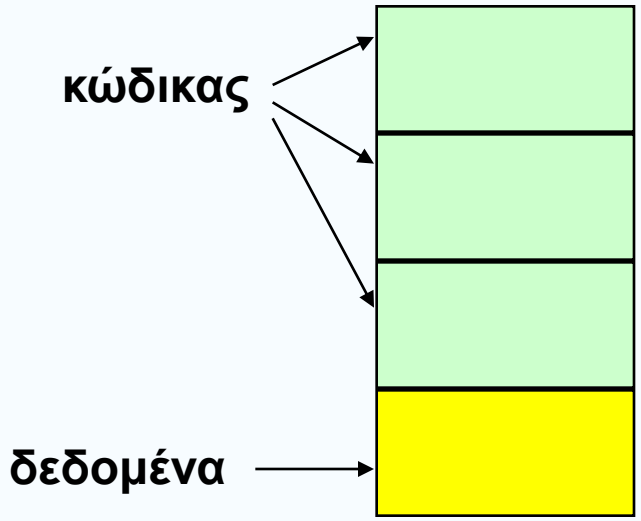


Παράδειγμα EAT

- Βασικές παραδοχές
  - Memory access time: 60 ns
  - TLB access time: 6 ns
- EAT χωρίς σελιδοποίηση: 60 ns
- EAT με χρήση του πίνακα σελίδων: 60 ns + 60 ns = 120 ns
- EAT TLB/πίνακας σελίδων
  - hit ratio 90%
    - EAT = 0.9 * (6 + 60) + 0.1 * (6 + 60 + 60)
      = 59.4 + 12.6 = 72
    - 20% απώλεια απόδοσης σε σχέση με τη μη σελιδοποίηση
  - hit ratio 99%
    - EAT = 0.99 * (6 + 60) + 0.01 * (6 + 60 + 60)
      = 65.34 + 1.26 = 66.60
    - 11% απώλεια απόδοσης σε σχέση με τη μη σελιδοποίηση



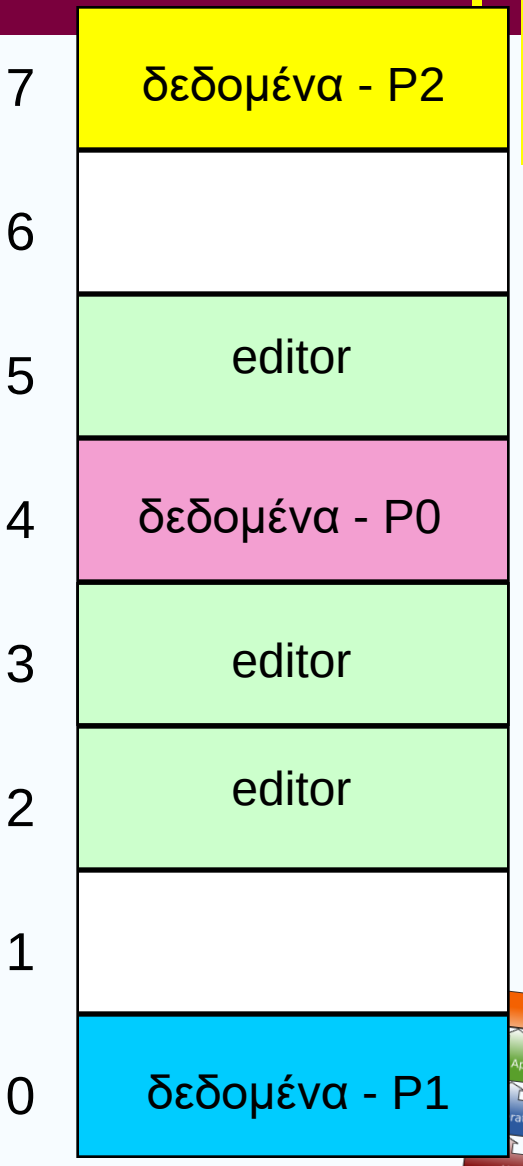
Διαμοιραζόμενες σελίδες

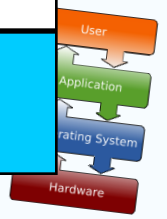


Διεργασία editor

Πίνακες σελίδων

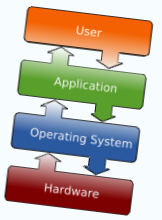
P0	P1	P2
5	5	5
2	2	2
3	3	3
4	0	7





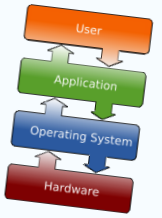
Διαμοιραζόμενος και ιδιωτικός κώδικας

- Διαμοιραζόμενος κώδικας
  - Μόνο για ανάγνωση επανεισαγόμενος (reentrant) διαμοιραζόμενος κώδικας μεταξύ των διεργασιών
  - Ο διαμοιραζόμενος κώδικας εμφανίζεται στην ίδια θέση στο φυσικό χώρο διευθύνσεων
- Ιδιωτικός κώδικας και δεδομένων
  - Κάθε διεργασία διατηρεί ένα ξεχωριστό αντίγραφο του κώδικα και των δεδομένων.
  - Οι ιδιωτικές σελίδες μπορούν να εμφανίζονται οπουδήποτε στο φυσικό χώρο διευθύνσεων.



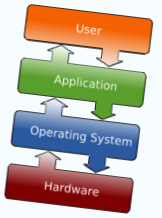
Κατάτμηση (Segmentation)

- Ένα τμήμα (segment) είναι ένα μεταβλητού μεγέθους σύνολο συνεχόμενων διευθύνσεων μνήμης στον ιδεατό χώρο διευθύνσεων μιας διεργασίας που οργανώνεται και διαχειρίζεται από το ΛΣ ως μια ενιαία μονάδα.
- Κατάτμηση είναι ο τρόπος οργάνωσης της ιδεατής μνήμης σε τμήματα.
- Μια διεύθυνση αποτελείται από δύο μέρη – έναν αριθμό τμήματος και μια μετατόπιση (offset).



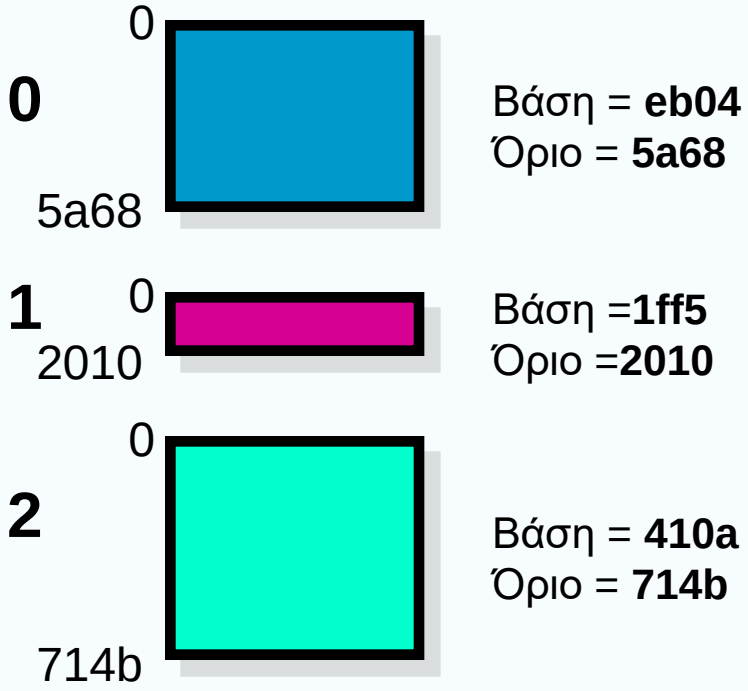
Κατάτμηση (συν.)

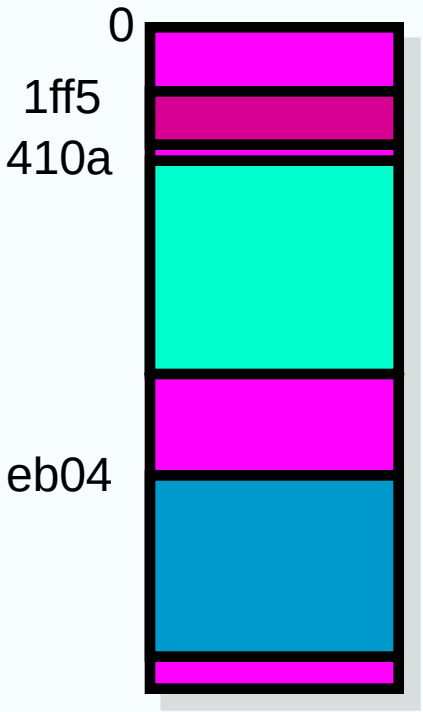
- Τα τμήματα δεν είναι ίσα και η κατάτμηση είναι παρόμοια με τη δυναμική τμηματοποίηση -> **Μειώνεται ο εσωτερικός κατακερματισμός**
- Τα τμήματα μπορούν να έχουν δυναμικό μέγεθος ώστε να απλοποιείται η διαχείριση δυναμικών δομών δεδομένων
- Η κατάτμηση :
  - Επιτρέπει στα προγράμματα να τροποποιούνται και να μεταφράζονται εκ νέου ανεξάρτητα
  - Είναι κατάλληλη για διαμοίραση και προστασία δεδομένων



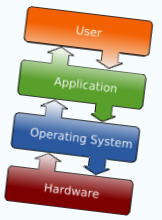
Κατάτμηση (συν.)

Μια εικονική διεύθυνση είναι ένας αριθμός τμήματος (**segment number)** και μια **μετατόπιση offset**.

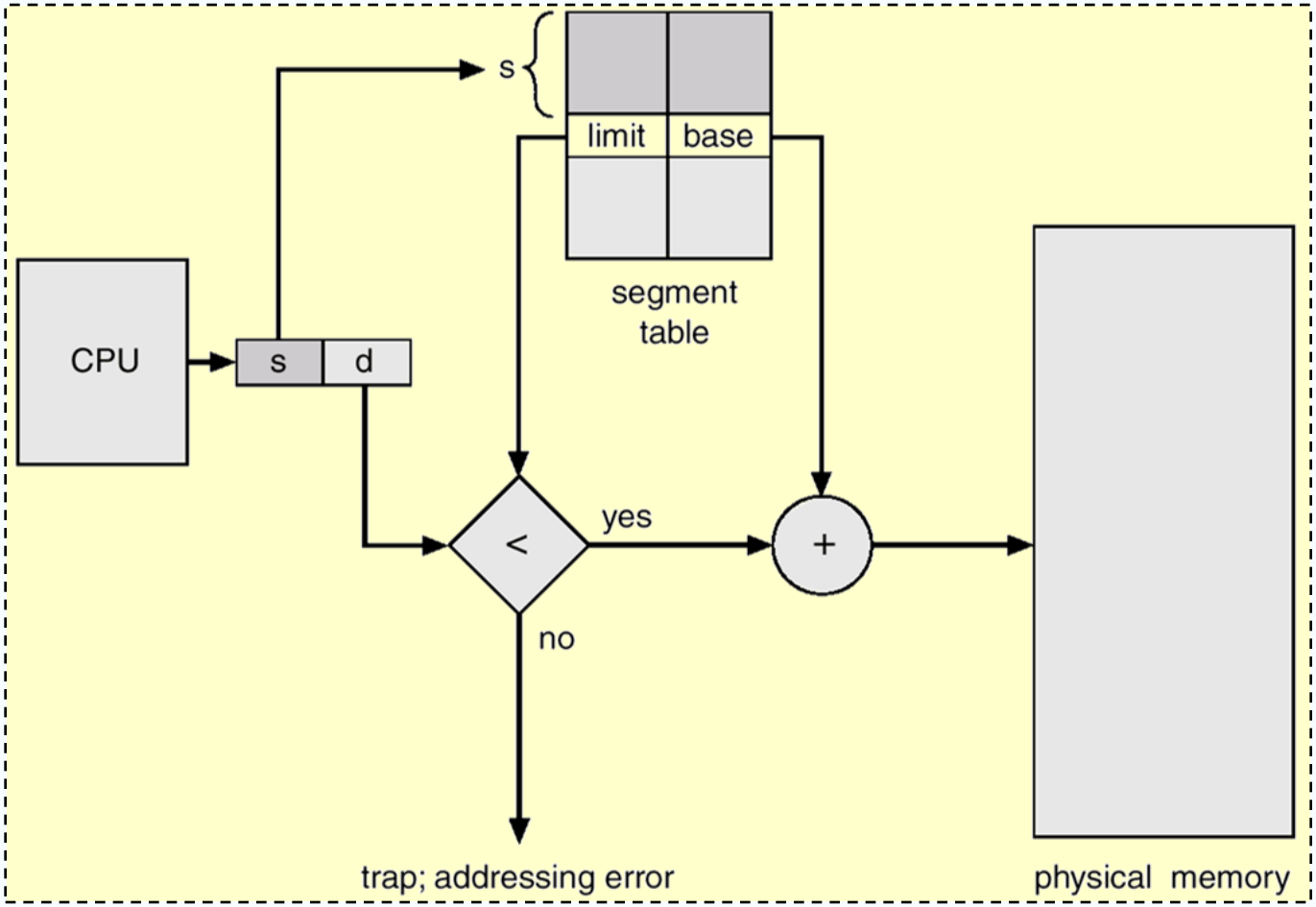


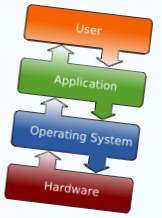


Κάθε τμήμα τοποθετείται σε μια συνεχόμενη περιοχή της μνήμης.



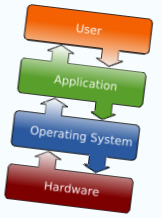
Μετατροπή διεύθυνσης σε σύστημα με κατάτμηση (1)





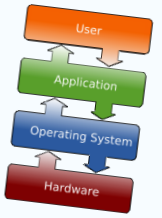
Κατάτμηση και προγραμματισμός

- Η κατάτμηση είναι φανερή στον προγραμματιστή σε αντίθεση με τη σελιδοποίηση, και παρέχεται ως διευκόλυνση για την οργάνωση προγραμμάτων και δεδομένων (αρκεί ο προγραμματιστής να γνωρίζει τα όρια τμημάτων που αναγνωρίζει η γλώσσα προγραμματισμού, καθώς και αν όλα μπορούν να έχουν δυναμικό μέγεθος!!!).
- Ο προγραμματιστής βλέπει το πρόγραμμα σαν συλλογή από segments π.χ. main program, function, objects, global variables, stack…
- **Δεν υπάρχει μια απλή συσχέτιση μεταξύ των λογικών και των φυσικών διευθύνσεων.**

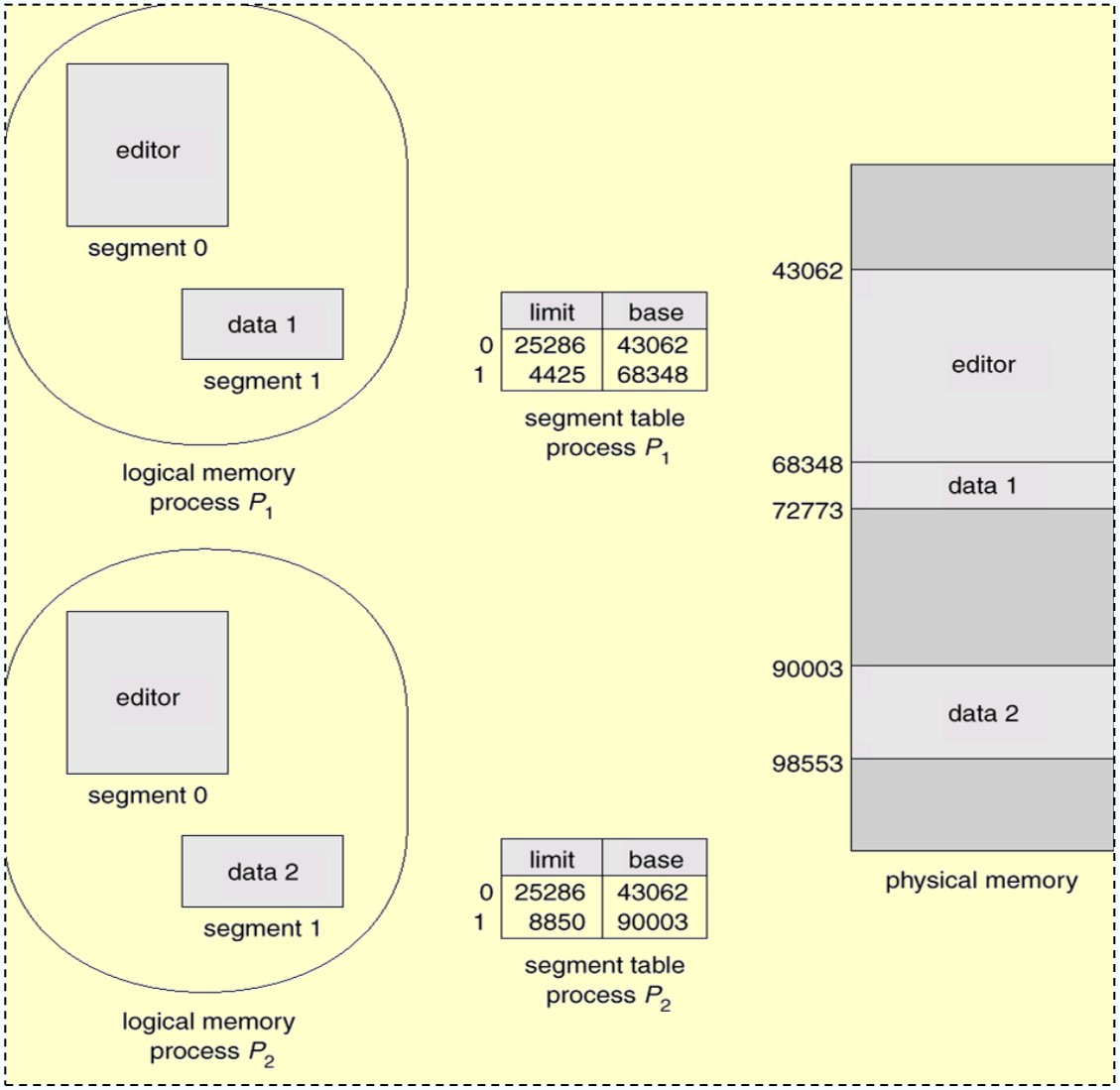


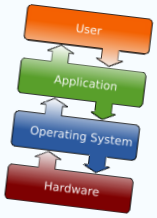
Πλεονεκτήματα της κατάτμησης

- Η εικόνα της μνήμης είναι η εικόνα που έχει ο προγραμματιστής (ή ο έμπειρος χρήστης)
- Τα τμήματα προστατεύονται μεταξύ τους
  - Κάθε τμήμα περιέχει ένα τύπο πληροφορίας (εντολές, στοίβα …)
- Η διαμοίραση τμημάτων είναι λογική και εύκολη
  - Αν όλες οι εντολές είναι σε ένα τμήμα και όλα τα δεδομένα σε άλλο, το τμήμα εντολών μπορεί να διαμοιραστεί ελεύθερα σε διαφορετικές διεργασίες (κάθε μια με τα δικά της δεδομένα)

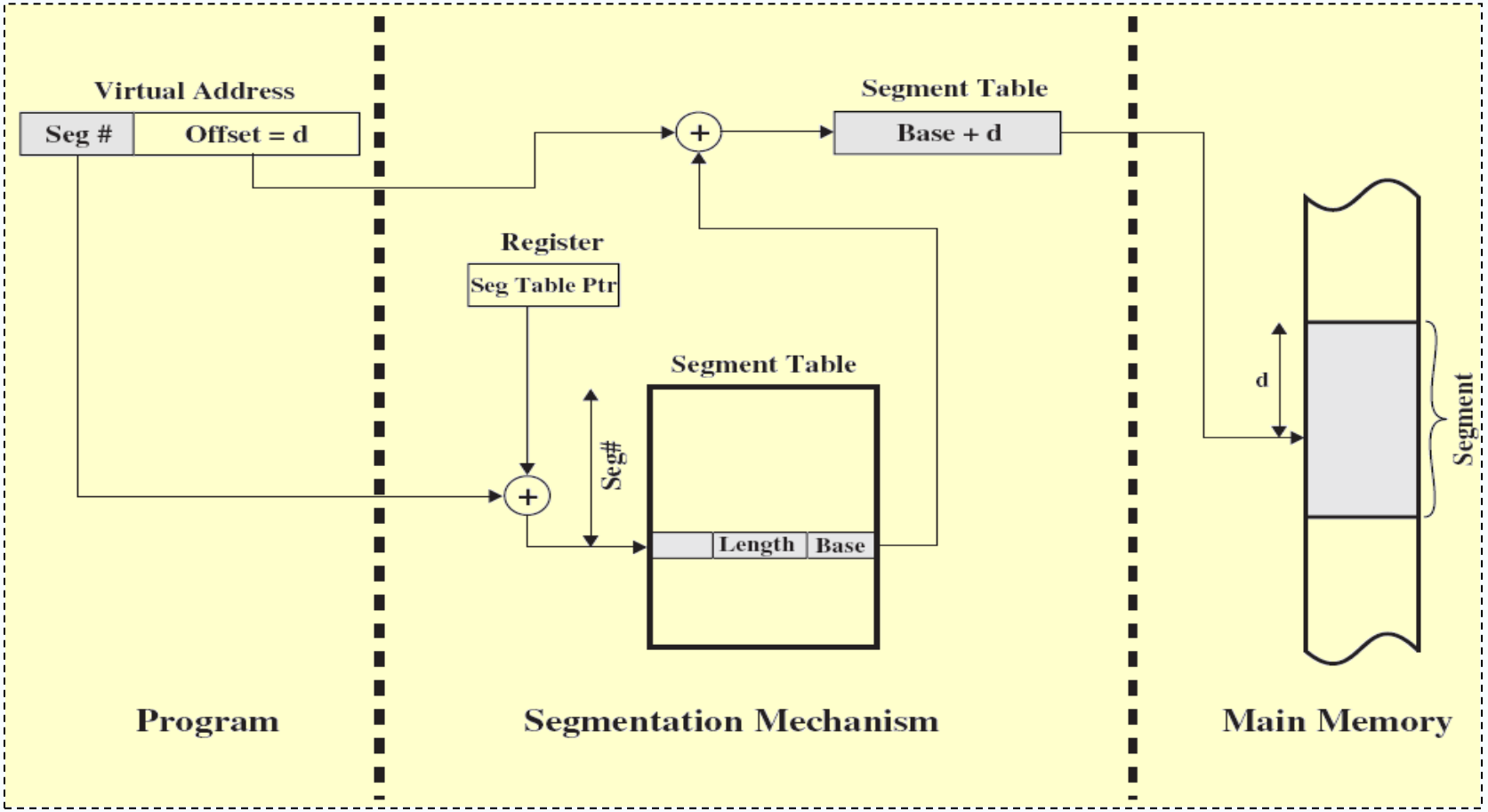


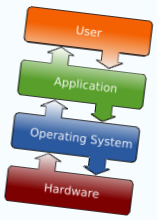
Διαμοίραση τμημάτων



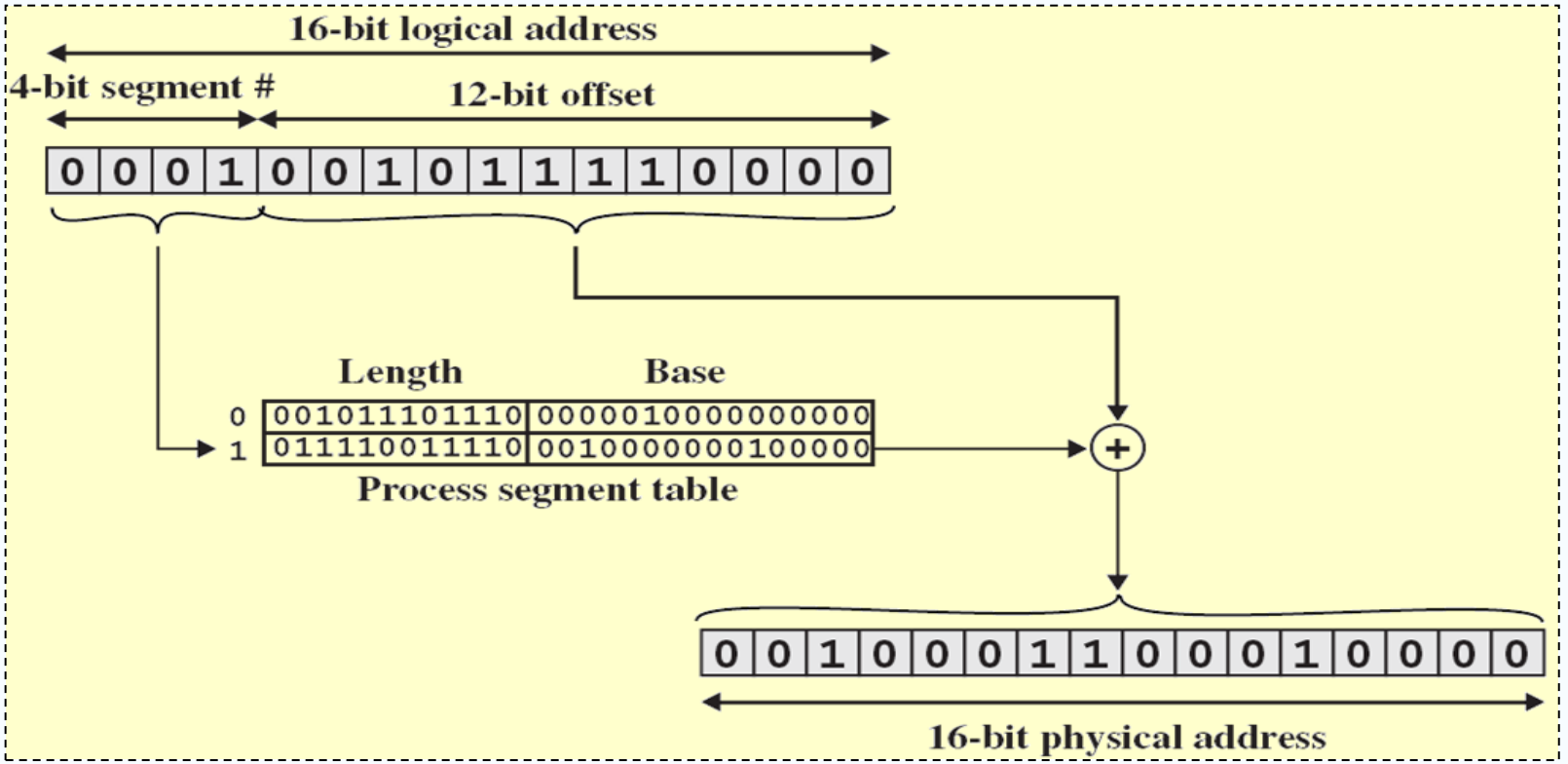


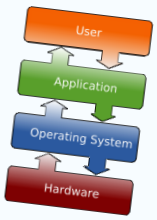
Μετατροπή διεύθυνσης σε σύστημα με κατάτμηση (2)





Κατάτμηση: μετατροπή λογικών σε φυσικές διευθύνσεις

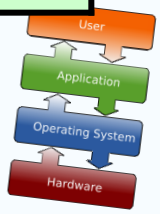




Ανάλυση του παραδείγματος

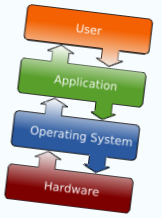
- Λογική διεύθυνση : **0001001011110000**
- Αριθμός τμήματος : 4 bits και είναι ο αριθμός **0001**
- Μετατόπιση τμήματος 12 bits
- Μέγιστο μέγεθος τμήματος 2^{12}=4096 bytes
- Η μετατόπιση της λογικής διεύθυνσης αντιστοιχεί στον αριθμό :
  $512+128+64+32+16=(752)_{10}$

2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	1	0	1	1	1	1	0	0	0	0



Μετάφραση σε φυσική διεύθυνση

1. Αριθμός τμήματος 0001
2. Ο αριθμός αυτός χρησιμοποιείται στον segment table της διεργασίας για να βρεθεί η φυσική διεύθυνση της αρχής του τμήματος
3. Συγκρίνεται η μετατόπιση με το μήκος του τμήματος. Αν ΜΕΤΑΤΟΠΙΣΗ > ΜΗΚΟΣ ΤΜΗΜΑΤΟΣ προκύπτει ΜΗ ΕΓΚΥΡΗ ΔΙΕΥΘΥΝΣΗ
4. **ΦΥΣΙΚΗ ΔΙΕΥΘΥΝΣΗ = ΑΘΡΟΙΣΜΑ : ΦΥΣΙΚΗΣ ΔΙΕΥΘΥΝΣΗΣ ΑΡΧΗΣ ΤΜΗΜΑΤΟΣ + ΜΕΤΑΤΟΠΙΣΗ**



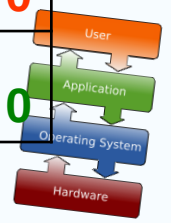
- Από τον process segment table προκύπτει ότι το **segment 0** έχει μήκος : 512 + 128 + 64 + 32 + 8 + 4 + 2 = 750

2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	1	0	1	1	1	0	1	1	1	0

•Άρα η λογική διεύθυνση πράγματι βρίσκεται στο segment 1 (επειδή 752 > 750)

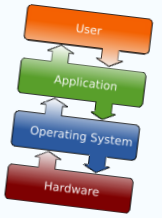
•Για να βρεθεί η φυσική διεύθυνση θα προσθέσω τη μετατόπιση στη βάση του segment 1 :

base	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0
offset	0	0	0	0	0	0	1	0	1	1	1	1	0	0	0	0
address	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	0

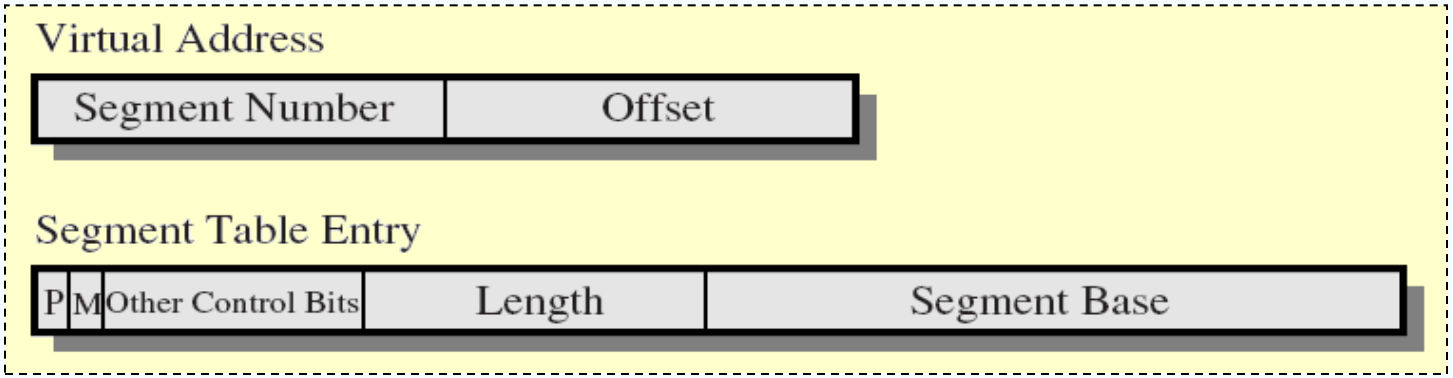


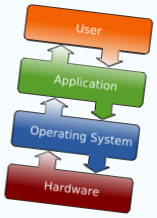
Πίνακες Τμημάτων (Segment Tables)

- Αντιστοιχίζουν τα segments στην κύρια μνήμη
- Κάθε είσοδος περιέχει τον αριθμό και το μήκος του segment
- Απαιτούνται :
  - ένα bit για να αποφασιστεί αν το segment είναι ήδη στην κύρια μνήμη
  - ένα επιπλέον bit για να αποφασιστεί αν το segment έχει μεταβληθεί από τότε που φορτώθηκε στην κύρια μνήμη
- Ο πίνακας τμημάτων έχει μεταβλητό μέγεθος, βρίσκεται στην κύρια μνήμη και ένας καταχωρητής κρατά την αρχική διεύθυνση του πίνακα τμημάτων για τη διεργασία που εκτελείται



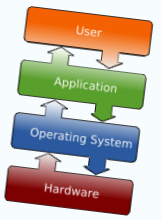
Είσοδοι Πίνακα Τμημάτων





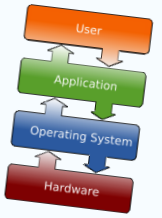
Η αρχιτεκτονική της κατάτμησης

- Ο compiler δημιουργεί segments για : main( ), functions, globals, …
  - Κάθε τμήμα αποθηκεύεται ξεχωριστά στη μνήμη
  - Πρέπει να υπάρχει υποστήριξη υλικού για την αντιστοίχιση λογικών διευθύνσεων (segment number + offset) σε φυσικές διευθύνσεις
- Καταχωρητές που χρησιμοποιούνται
  - **segment table base register (STBR) :** δείχνει στη διεύθυνση μνήμης του πίνακα τμημάτων στη μνήμη
  - **segment table length (limit) register (STLR):** δείχνει τον αριθμό των τμημάτων που χρησιμοποιεί το πρόγραμμα



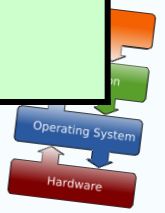
Σύγκριση απλής κατάτμησης και σελιδοποίησης

- Η κατάτμηση απαιτεί περισσότερο σύνθετο hardware για τον μετασχηματισμό των διευθύνσεων
- Η κατάτμηση έχει το μειονέκτημα του εξωτερικού κατακερματισμού
- Η σελιδοποίηση δημιουργεί πολύ μικρό εσωτερικό κατακερματισμό
- Η κατάτμηση είναι ορατή από τον προγραμματιστή ενώ η σελιδοποίηση είναι αδιαφανής
- Η κατάτμηση θεωρείται ως πλεονέκτημα που προσφέρεται στον προγραμματιστή για να οργανώσει λογικά ένα πρόγραμμα σε segments και να χρησιμοποιήσει διαφορετικά είδη προστασίας (π.χ. execute-only, read-write)
  - Για το σκοπό αυτό στους πίνακες τμημάτων πρέπει να χρησιμοποιούνται bits προστασίας



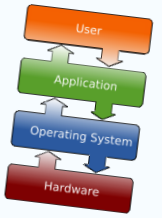
Σύγκριση σελιδοποίησης και κατάτμησης

	Σελιδοποίηση	Κατάτμηση
Είναι απαραίτητο να γνωρίζει ο προγραμματιστής ότι χρησιμοποιείται αυτή η τεχνική;	ΟΧΙ	ΝΑΙ
Πόσοι χώροι γραμμικών διευθύνσεων υπάρχουν;	1	ΠΟΛΛΟΙ
Ο συνολικός χώρος διευθύνσεων υπερβαίνει το μέγεθος της φυσικής μνήμης;	ΝΑΙ	ΝΑΙ
Μπορούν οι διαδικασίες και τα δεδομένα να διαχωριστούν και να προστατευθούν ξεχωριστά;	ΟΧΙ	ΝΑΙ
Μπορούν πίνακες με αυξομειούμενο μέγεθος να εξυπηρετηθούν εύκολα;	ΟΧΙ	ΝΑΙ
Διευκολύνεται η διαμοίραση των διαδικασιών μεταξύ των χρηστών;	ΟΧΙ	ΝΑΙ
Ποιος είναι ο σκοπός αυτής της τεχνικής;	Η απόκτηση ενός μεγάλου γραμμικού χώρου διευθύνσεων χωρίς να είναι αναγκαία η αγορά επιπλέον φυσικής μνήμης	Να δοθεί η δυνατότητα σε προγράμματα και δεδομένα να διασπαστούν σε ανεξάρτητες λογικές ενότητες που διαμοιράζονται και προστατεύονται εύκολα.

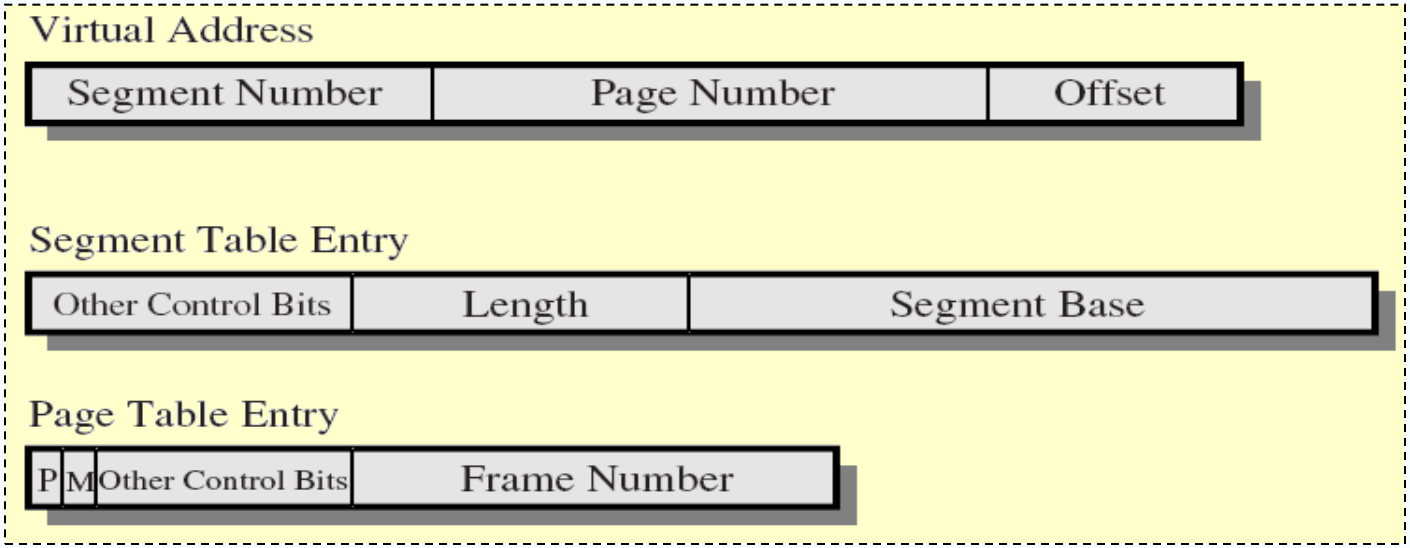


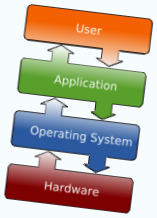
Κατάτμηση με σελιδοποίηση

- Οι σύγχρονοι υπολογιστές χρησιμοποιούν συνδυασμό κατάτμησης και σελιδοποίησης
- Ο χώρος διευθύνσεων του χρήστη χωρίζεται σε ένα πλήθος τμημάτων και κάθε τμήμα διασπάται σε ένα πλήθος σελίδων σταθερού μεγέθους που είναι το ίδιο με το μέγεθος πλαισίου της κύριας μνήμης.
- Κάθε διεργασία συνδέεται με έναν πίνακα τμήματος και έναν αριθμό από πίνακες σελίδων, έναν για κάθε τμήμα της διεργασίας.
- Επιλύεται το πρόβλημα του εξωτερικού κατακερματισμού μέσω της σελιδοποίησης των τμημάτων
- Κάθε είσοδος στον πίνακα τμημάτων δεν περιέχει τη διεύθυνση βάσης του τμήματος, αλλά τη διεύθυνση βάσης του πίνακα σελίδων που αντιστοιχεί στο τμήμα.

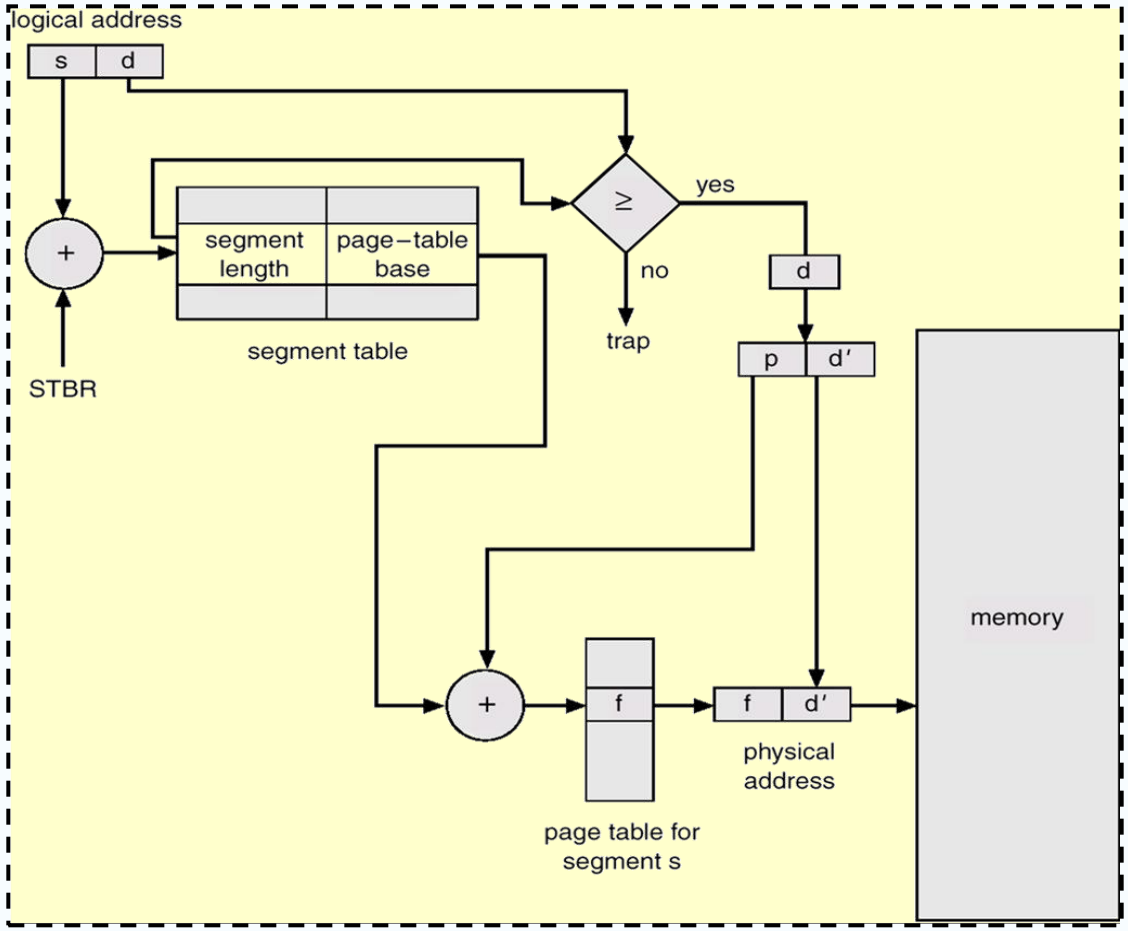


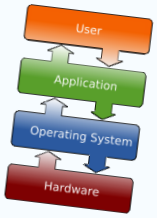
Είσοδοι πινάκων τμημάτων και σελίδων



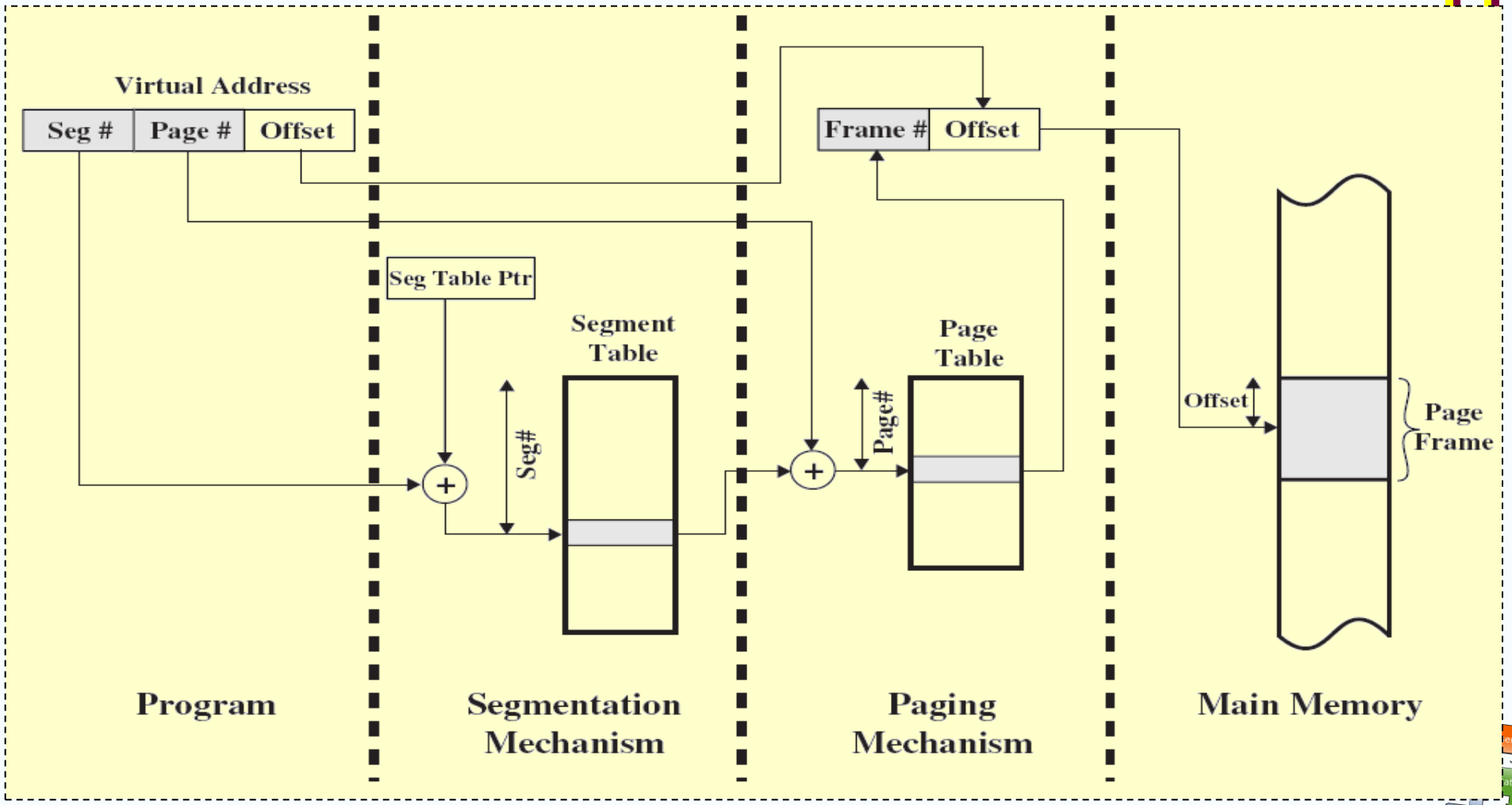


Μετατροπή διεύθυνσης (1)



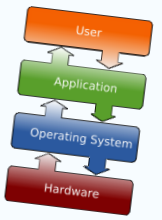


Μετατροπή διεύθυνσης (2)



Άσκηση 8.1

- Ποιο είναι το μεγαλύτερο μέγεθος προγράμματος που χωρά σε ένα πίνακα σελίδων σε ένα σύστημα που χρησιμοποιεί φυσικές διευθύνσεις 32-bit και μέγεθος σελίδας 1K;
- **ΛΥΣΗ**
  - ΦΥΣ. ΔΙΕΥΘΥΝΣΗ 32 bits→ χώρος διευθύνσεων **2^{32} διαφορετικές διευθύνσεις**
  - Μέγεθος σελίδας 1K= **2^{10} διευθύνσεις μνήμης**
  - ΑΡΑ : μέγιστο πλήθος σελίδων = **$2^{32}/2^{10}=2^{22}$**
  - Κάθε διεύθυνση των 32 bits (=4 bytes) πρέπει να χωρά σε κάθε μια από τις 2^{22} θέσεις του πίνακα σελίδων άρα μέγεθος του πίνακα σελίδων : **2^{22} X 4bytes = 16Mb**
  - μεγαλύτερο μέγεθος προγράμματος : **16Mb**

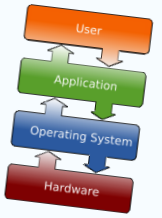


Άσκηση 8.2

- Δίνεται ο πίνακας σελίδων μιας διεργασίας :

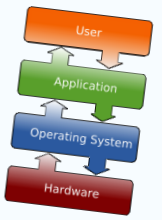
1. Ποιο είναι το μικρότερο δυνατό μέγεθος σελίδας;
2. Δώστε σε δυαδική μορφή τη φυσική διεύθυνση της εικονικής διεύθυνσης 1234
3. Δώστε σε δεκαδική μορφή τη φυσική διεύθυνση της εικονικής διεύθυνσης 3333

Page No	Frame No
0	5
1	3
2	4
3	0
4	1
5	2

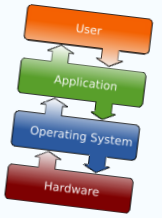


Λύση Άσκησης 8.2

- Γιατί το μέγεθος σελίδας είναι 1K;
  - Έστω μέγεθος σελίδας 512 bytes (=2^9 bytes)
    - Η εικονική διεύθυνση 3333 απαιτεί **6 γεμάτες σελίδες** (=512 X 6 =3072) και στην **7η σελίδα μετατόπιση** (offset) = 3333-3072=261
    - Άρα απαιτούνται αριθμοί σελίδων **0, 1, 2, 3, 4, 5, 6, 7**

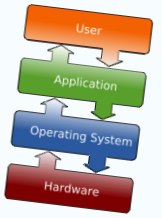


- Ο πίνακας όμως δεν έχει αριθμό σελίδας μεγαλύτερο του 5. Στην περίπτωση που θεωρηθεί ως μέγεθος σελίδας 512 bytes αυτομάτως η διεύθυνση 3333 είναι εκτός μνήμης, ΑΔΥΝΑΤΟ ΔΙΟΤΙ μας δίνεται ότι ο πίνακας σελίδων της διεργασίας υφίσταται.
- ΕΠΟΜΕΝΩΣ Ο ΕΠΟΜΕΝΟΣ ΑΡΙΘΜΟΣ ΠΟΥ ΥΠΑΡΧΕΙ ΩΣ ΔΥΝΑΜΗ ΤΟΥ 2 ΕΊΝΑΙ 2^{10} =1024 bytes=1KB



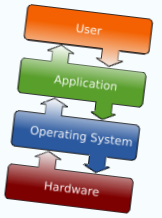
Άσκηση – 8.3

- Θεωρείστε ένα λογικό χώρο διευθύνσεων που αποτελείται από οκτώ σελίδες με μέγεθος κάθε σελίδας 1024 bytes, που αντιστοιχείται σε μια φυσική μνήμη που αποτελείται από 32 frames.
- Πόσα bits υπάρχουν στη λογική διεύθυνση?
- Πόσα bits υπάρχουν στη φυσική διεύθυνση?



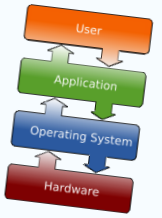
Άσκηση – 8.4

- Θεωρείστε ένα σύστημα που χρησιμοποιεί :
  - απλή σελιδοποίηση και
  - τεχνική TLB
- Αν μια αναφορά στη μνήμη απαιτεί 400ns, μια αναφορά στο TLB απαιτεί 50ns και το ποσοστό επιτυχίας (hit - rate) στο TLB είναι 80% ποιος είναι ο πραγματικός χρόνος αναφοράς στη μνήμη; Πόση είναι η βελτίωση στην ταχύτητα (speed-up) λόγω χρήσης της τεχνικής TLB;



Λύση άσκησης 8.4

- Απλή σελιδοποίηση : Κάθε αναφορά στη μνήμη απαιτεί 2 προσπελάσεις άρα συνολικός χρόνος : 400ns + 400ns = 800 ns
- Απλή σελιδοποίηση και TLB :
  - Επιτυχία (hit) στο TLB : 50ns + 400ns = 450ns
  - Αποτυχία (miss) στοTLB : 50ns + 400ns + 400ns = 850ns
- Πραγματικός χρόνος προσπέλασης με απλή σελιδοποίηση και TLB :
  - 450*80% + 850*20% = 530ns
- Speed-up = 800/530 = 1.51



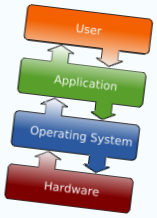
Άσκηση – 8.5

- Σε μια αρχιτεκτονική χρησιμοποιούνται λογικές διευθύνσεις των 32 bits χωρισμένες ως εξής :

4-bit segment number **|** ***12-bit page number*** **|** ***16-bit offset***

Ποιο είναι το μέγεθος σελίδας;

Ποιο είναι το μέγιστο μήκος τμήματος;



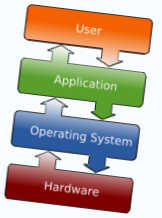
Άσκηση – 8.6

- Θεωρείστε τον ακόλουθο πίνακα τμημάτων:

τμήμα	βάση	μήκος
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Ποιες φυσικές διευθύνσεις αντιστοιχούν στις παρακάτω λογικές διευθύνσεις;

i. **0,430**
ii. **1,10**
iii. **2,500**
iv. **3,400**
v. **4,112**



Αναφορές

- “Λειτουργικά Συστήματα – Αρχές Σχεδίασης”, 4η έκδοση, W. Stallings, Εκδόσεις Τζιόλα, 2008.
- “Operating System Concepts”, 7η έκδοση, από Abraham Silberschatz, Peter Galvin και Greg Gagne, Addison-Wesley, 2004.
- “Operating Systems: Design and Implementation”, 3η έκδοση, από Andrew Tanenbaum και Albert Woodhull, Prentice Hall, 2006.