ΗΥ 232
Οργάνωση και Σχεδίαση Υπολογιστών

Διάλεξη 16

Εικονική Μνήμη (Virtual Memory)

Νίκος Μπέλλας
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Απλό πείραμα

```
int *data = malloc((1<<20) * sizeof(int));

printf("Data[.] starts @ 0x%lx\n", data);
```

```
% a.out
  Data[.] starts @ 0x7fc7fb798010
% cat /proc/meminfo | grep MemTotal
  MemTotal:         8185684 kB
```

- Ο υπολογιστής μου έχει 8 GB μνήμη.
- Πως είναι δυνατόν το array *data[.]* να αποθηκεύεται πέρα από την μνήμη που έχουμε? (στα >140 TB)

Όταν προγραμματίζετε, χρησιμοποιείτε εικονική μνήμη (virtual memory)

Προβλήματα φυσικής μνήμης (1)

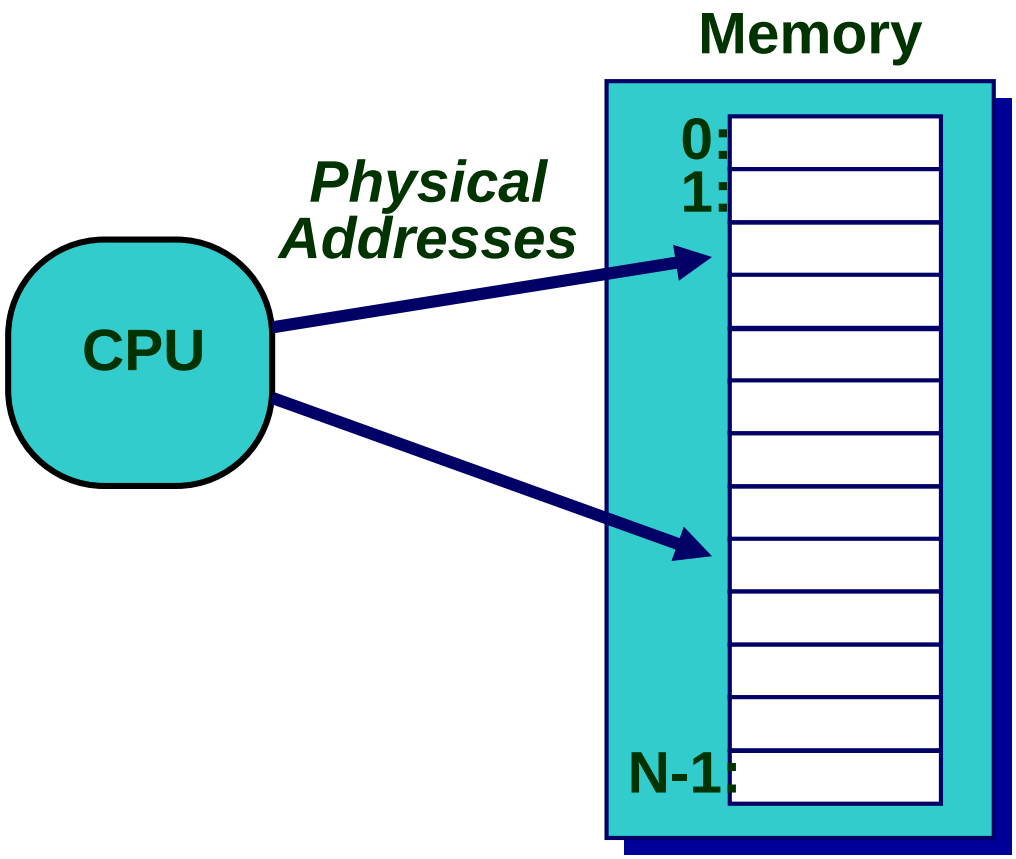
- Τι θα συμβεί εάν η μνήμη που χρειάζεται το πρόγραμμά μας είναι περισσότερη από το μέγεθος της κύριας μνήμης;
  - Θα μπορούσαμε να αποθηκεύσουμε όλο το πρόγραμμα στον σκληρό δίσκο, και να χρησιμοποιήσουμε την κύρια μνήμη σαν cache του σκληρού δίσκου
  - Αυτό είναι η εικονική μνήμη (virtual memory): μια τεχνική που επιτρέπει την χρησιμοποίηση της κύριας μνήμης ως cache του σκληρού δίσκου.
  - Πριν την εφεύρεση της εικονικής μνήμης, οι προγραμματιστές έπρεπε να προνοήσουν να μεταφέρουν προγράμματα και δεδομένα μεταξύ δίσκου και κύριας μνήμης κάθε φορά που αυτά δεν χωρούσαν στην κύρια μνήμη.
    - Πολύπλοκο, κουραστικό και γεμάτο κινδύνους για λάθη. Οι προγραμματιστές παλαιότερα έπρεπε να κάνουν πολύ περισσότερα από το απλώς να γράφουν ένα πρόγραμμα για μια εφαρμογή.

Προβλήματα φυσικής μνήμης (2)

- Τι θα συμβεί εάν θέλουμε να εκτελούμε σε έναν επεξεργαστή ταυτόχρονα πολλά προγράμματα ή διεργασίες (processes);
  1. Ακόμα και εάν ένα πρόγραμμα χωράει στην κύρια μνήμη, 10 προγράμματα μπορεί να μην χωράνε.
  2. Πολλαπλά προγράμματα θέλουν αν γράψουν στην ίδια διεύθυνση μνήμης; Πρόβλημα!
     - Τι θα συμβεί εάν δύο διαφορετικά προγράμματα Α και Β θέλουν πχ να χρησιμοποιήσουν την θέση μνήμης 0x10000000 ως βάση του stack τους;
  3. Πως μπορούμε να προστατεύσουμε τα δεδομένα ενός προγράμματος από το να διαβαστούν ή και να γραφούν από κάποιο άλλο πρόγραμμα;

Συστήματα με μόνο φυσική μνήμη

- Παραδείγματα στον πραγματικό κόσμο:
  - Πολλοί παλαιότεροι υπερυπολογιστές *Cray*
  - Πολλά ενσωματωμένα συστήματα (embedded systems)
- Πλήρης έλεγχος της μνήμης από τον προγραμματιστή



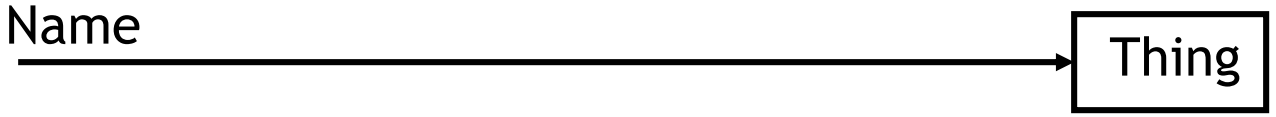
Εικονική Μνήμη (Virtual Memory)

- Βασική ιδέα:
  - Δώσε στον προγραμματιστή την ψευδαίσθηση ότι έχει ένα πολύ μεγάλο χώρο διευθύνσεων
  - Ενώ, στην πραγματικότητα έχει πολύ λιγότερη φυσική μνήμη
  - Ο προγραμματιστής δεν πρέπει να ασχολείται με την διαχείριση (management) της μνήμης

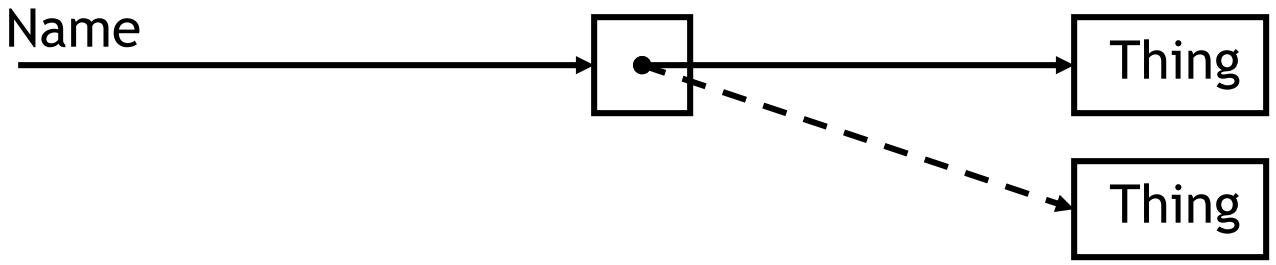
- Μηχανισμοί hardware + system software σε συνεργασία διαχειρίζονται την φυσική μνήμη για να δημιουργήσουν την ψευδαίσθηση στον προγραμματιστή.

Η ιδέα της έμμεσης προσπέλασης

- Η έμμεση προσπέλαση είναι ή δυνατότητα να μπορούμε να προσπελάσουμε ένα αντικείμενο (πχ μια μεταβλητή) χρησιμοποιώντας ένα συμβολικό όνομα αντί την τιμή του ίδιου του αντικειμένου. Αγγλικός όρος *indirection*.
- Άμεση προσπέλαση



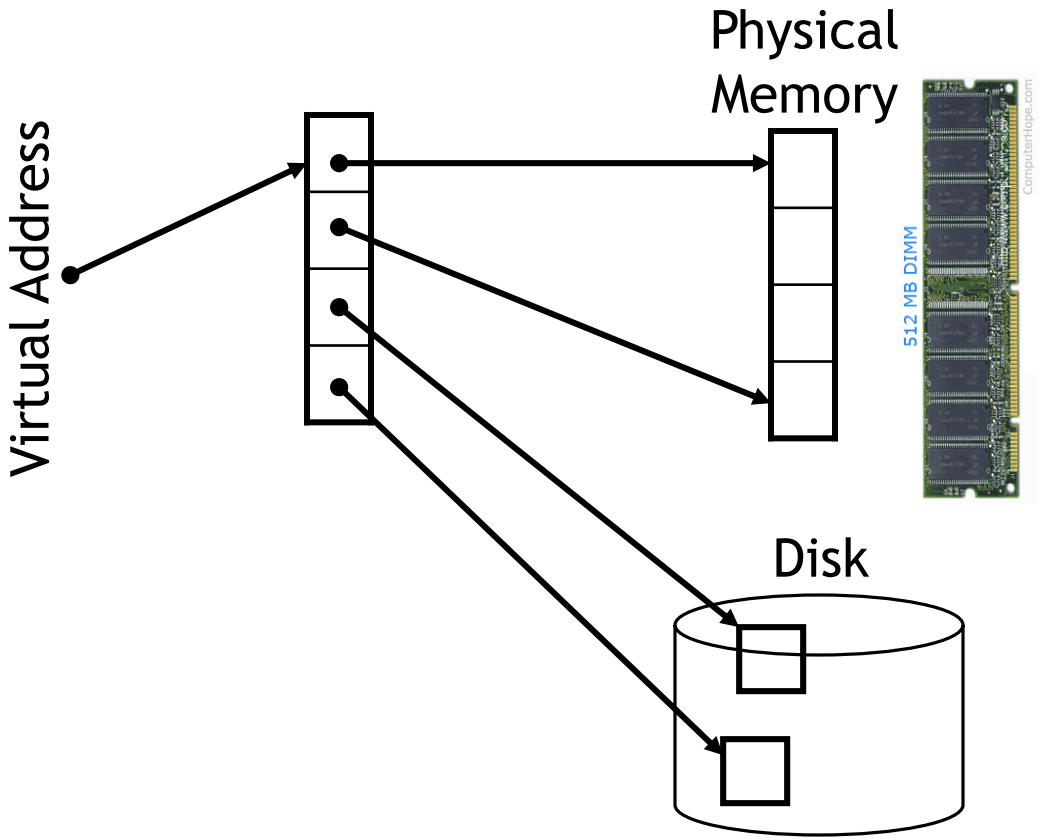
- Έμμεση προσπέλαση



- Παράδειγμα είναι οι pointers στην C

Εικονική Μνήμη (Virtual Memory)

- Οι εικονικές διευθύνσεις (virtual addresses) που χρησιμοποιείτε σαν προγραμματιστές μεταφράζονται σε φυσικές (δηλ. πραγματικές) διευθύνσεις (physical addresses) στην κύρια μνήμη ή στον σκληρό δίσκο.
- Η μετάφραση αυτή από εικονικές σε φυσικές διευθύνσεις υλοποιεί την ιδέα της έμμεσης προσπέλασης.



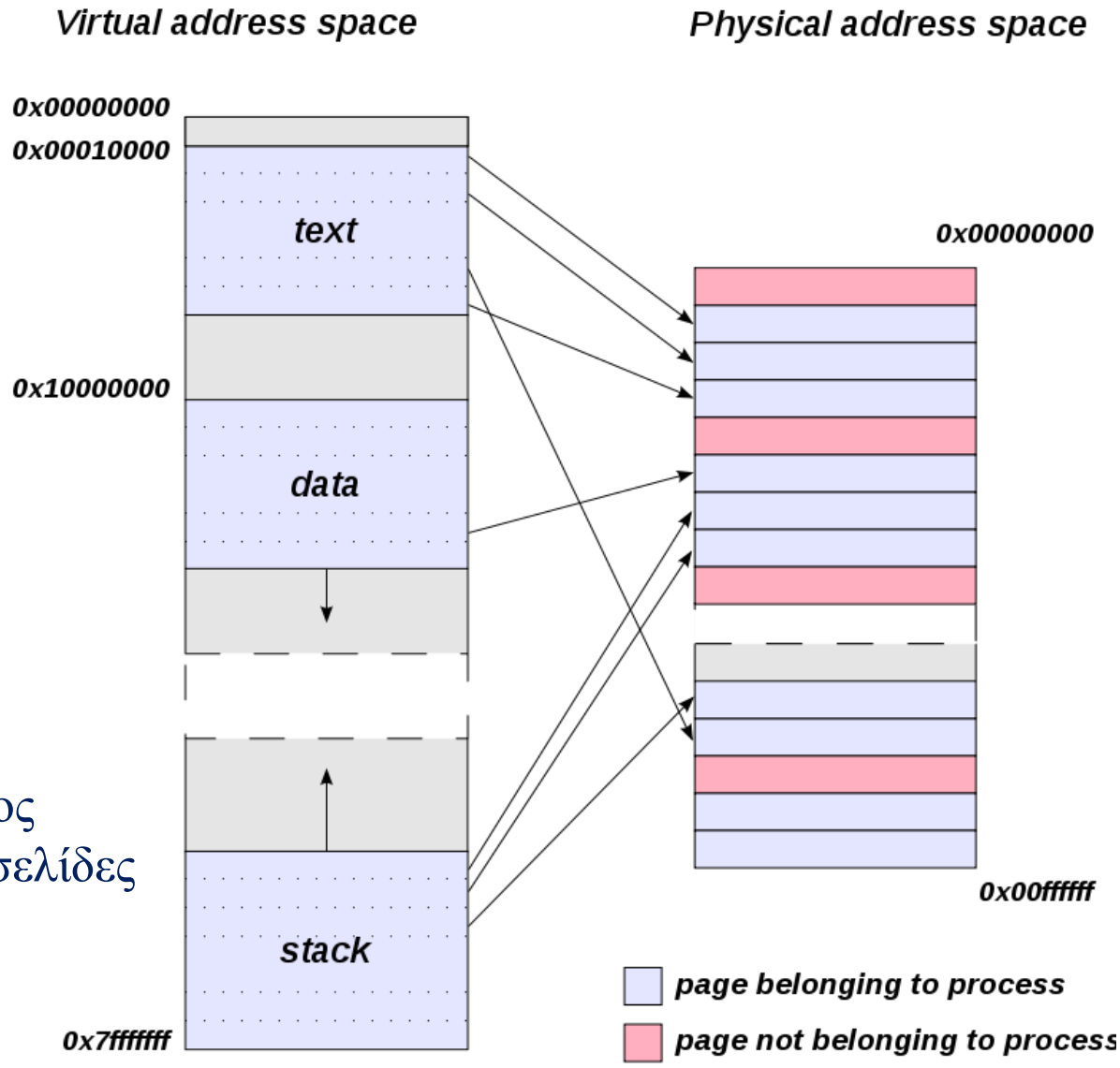
Δηλ., οι διευθύνσεις μεταβλητών που χρησιμοποιείτε στα προγράμματα σας είναι εικονικές, δεν αντιστοιχούν δηλ. σε πραγματικές διευθύνσεις στην μνήμη ή στον σκληρό δίσκο.

Πχ.

```
int *ptr = (int *)
0x1000FFFF;
```

Η διεύθυνση 0x1000FFFF είναι εικονική.

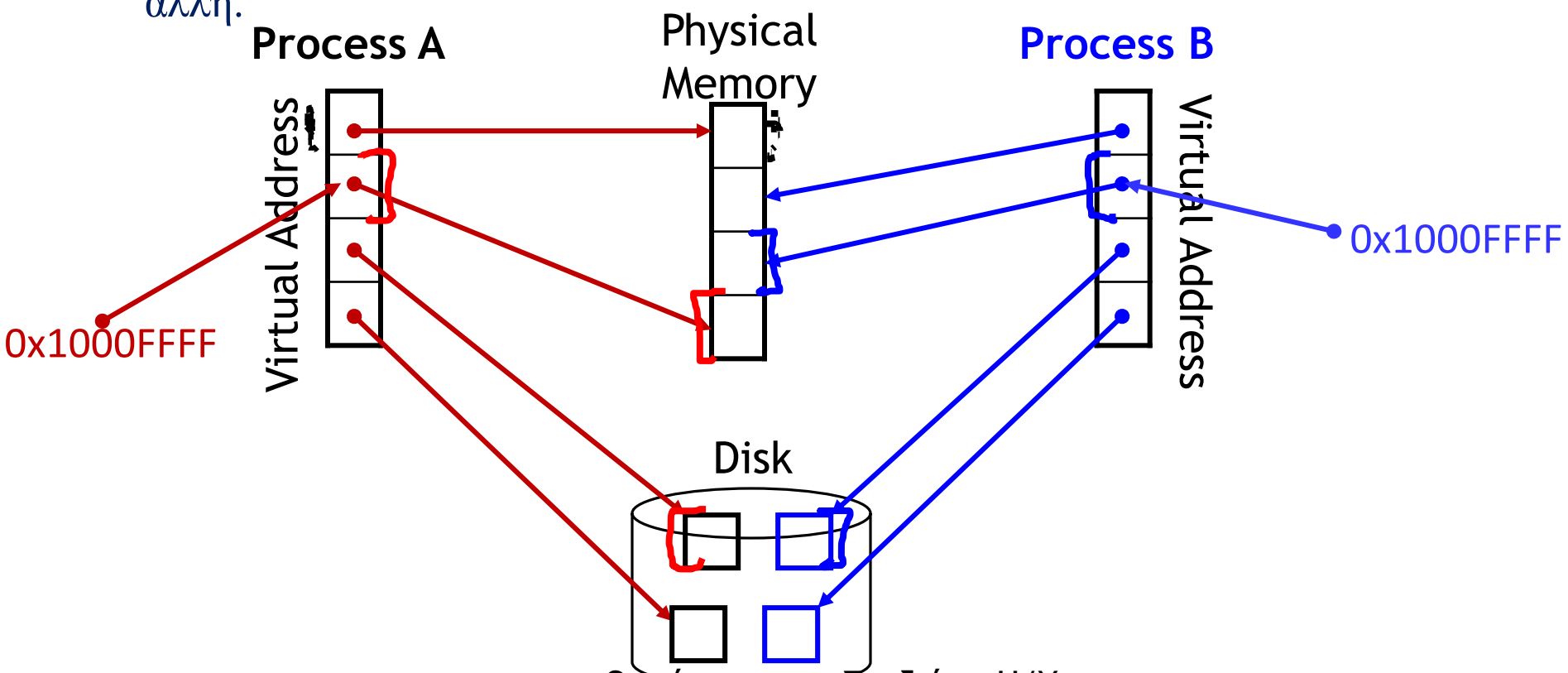
Εικονική Μνήμη (Virtual Memory)



Κάθε κομμάτι του προγράμματος απεικονίζεται σε διαφορετικές σελίδες στην φυσική μνήμη

Εικονική Μνήμη (Virtual Memory)

- Διαφορετικές διεργασίες έχουν διαφορετική απεικόνιση (mapping) από την εικονική στην φυσική μνήμη (virtual → physical).
- Αυτό έχει σαν αποτέλεσμα ότι ακόμα και να χρησιμοποιούν δύο διεργασίες την ίδια εικονική διεύθυνση, αυτή θα απεικονισθεί σε διαφορετικές φυσικές θέσεις μνήμης.
- Με το να αντιστοιχήσουμε διαφορετικές περιοχές φυσικής μνήμης στην διεργασία Α και Β, εμποδίζουμε το να διαβάζουν και να γράφουν η μία διεργασία από την άλλη.

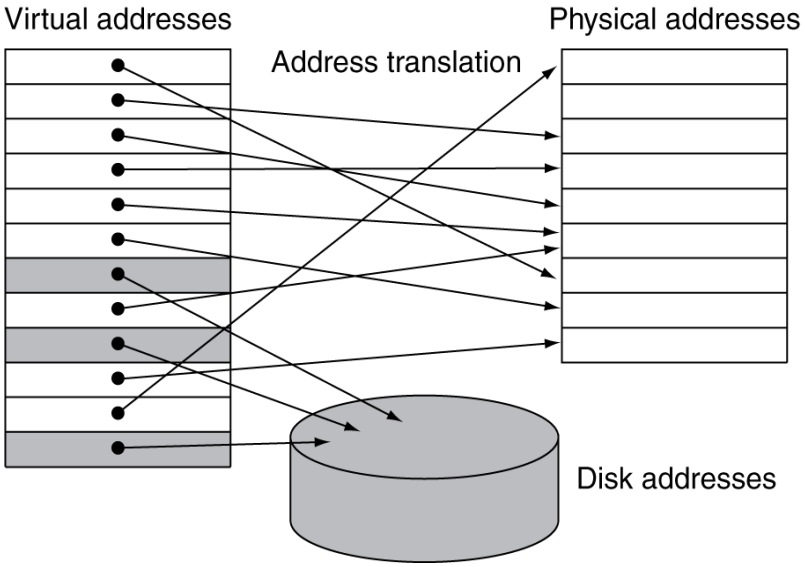


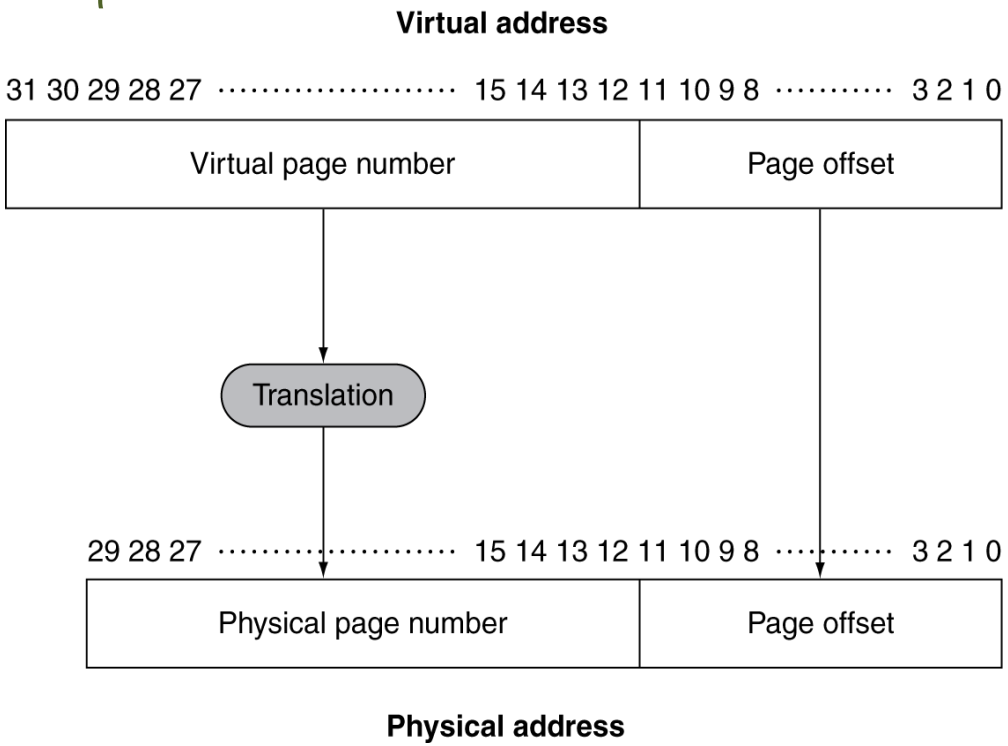
Εικονική μνήμη ως μηχανισμός cache

- Ο μηχανισμός μετάφρασης από εικονική σε φυσική μνήμη μπορεί να θεωρηθεί και ως μηχανισμός cache. Έχουμε τα ίδια προβλήματα να επιλύσουμε όπως και με την ιεραρχία μνήμης.
  - Θέλουμε να έχουμε το μέγεθος του σκληρού δίσκου και την ταχύτητα προσπέλασης της φυσικής μνήμης.
- Ο μηχανισμός της εικονικής μνήμης κρύβει τον μεγάλο χρόνο προσπέλασης του σκληρού δίσκου. Και εδώ το πράγματα είναι πολύ χειρότερα:
  - Ενώ ο χρόνος προσπέλασης της κύριας μνήμης είναι 200-400 κύκλοι μηχανής, ο μέσος χρόνος προσπέλασης του σκληρού δίσκου είναι περίπου δέκα εκατομμύρια κύκλοι!
  - Συνεπώς δεν επιτρέπονται πολλές αστοχίες κατά την μετάφραση

Μετάφραση εικονικής διεύθυνσης

- Η CPU και το λειτουργικό σύστημα μεταφράζουν τις εικονικές διευθύνσεις σε φυσικές.
  - Κάθε εικονική διεύθυνση χωρίζεται σε αριθμό σελίδας (virtual page number) και μετατόπιση σελίδας (page offset).
  - Οι σελίδες είναι το αντίστοιχο του block στις caches, πλην όμως πολύ μεγαλύτερες (αρκετά KB)
  - Μόνο ο αριθμός σελίδας μεταφράζεται. Η μετατόπιση είναι η ίδια τόσο στην εικονική όσο και στην φυσική διεύθυνση



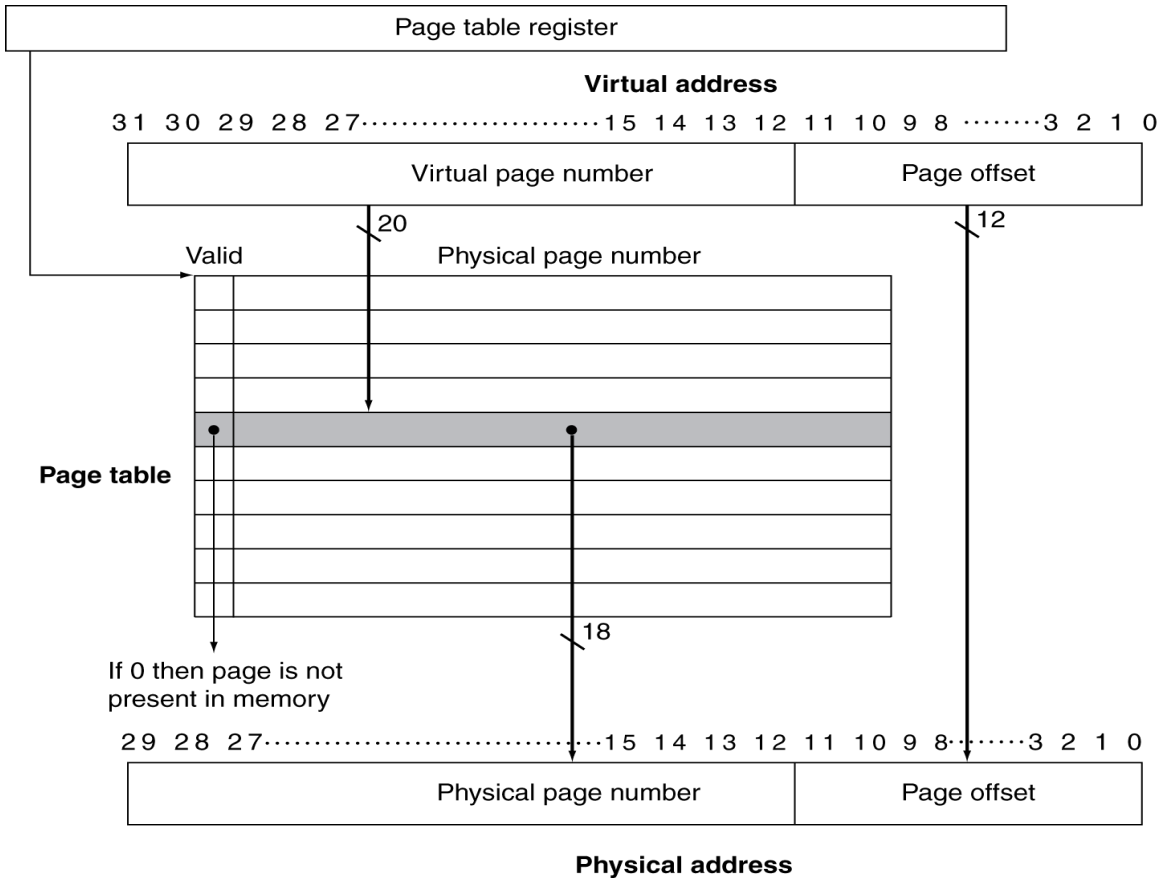


Πως βρίσκουμε την κατάλληλη σελίδα

- Η μετάφραση γίνεται με την χρήση του πίνακα σελίδων (page table).
  - **Κάθε διεργασία (process) έχει το δικό της πίνακα σελίδων**
  - Ο καταχωρητής πίνακα σελίδων (page table base register, PTBR) δείχνει την αρχή του πίνακα σελίδων για την διεργασία που είναι ενεργή αυτήν την στιγμή
  - Ο πίνακας σελίδων αποθηκεύει την φυσική διεύθυνση της αρχής της σελίδας
  - Επίσης και control bits το πιο σημαντικό εκ' των οποίων είναι το valid bit.

Πίνακας σελίδων και μετάφραση

- Κάθε γραμμή του πίνακα σελίδων περιέχει την διεύθυνση του πρώτου byte της φυσικής σελίδας, και το valid bit.
- Εάν valid bit = 1, η φυσική σελίδα είναι στην κύρια μνήμη.
- Εάν valid bit = 0, η φυσική σελίδα είναι στον σκληρό δίσκο και προκαλεί σφάλμα σελίδας (page fault). Παίρνει εκατομμύρια κύκλους μηχανής να φέρουμε την σελίδα από τον σκληρό δίσκο.



Σύστημα με 2^{20} σελίδες και μέγεθος σελίδας 4KB.
Ο πίνακας σελίδων έχει 2^{20} = 1M εισόδους (Page Table Entries, PTE).

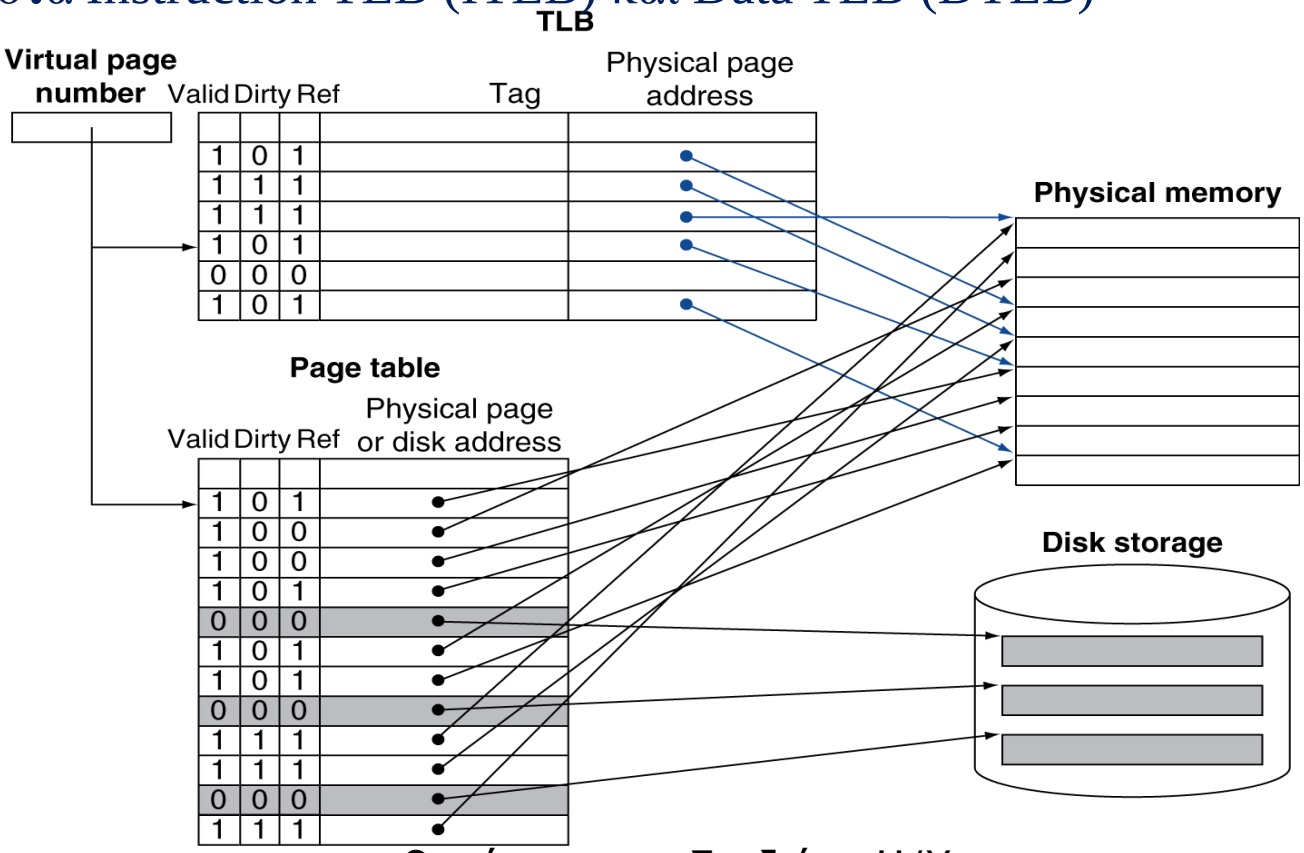
Υπάρχει όμως ένα πρόβλημα

- Για να κάνουμε την μετάφραση πρέπει να προσπελάσουμε τον πίνακα σελίδων στην κύρια μνήμη. Μετά γίνεται και η κανονική προσπέλαση.
- Δηλ. κάθε προσπέλαση μνήμης, είτε εντολές είτε δεδομένα, πρέπει να προσπελάσει πρώτα την κύρια μνήμη, **αχρηστεύοντας την όλη ιδέα της cache.**
- Και δεν έχουμε μιλήσει ακόμα για την χειρότερη περίπτωση, δηλ. η φυσική σελίδα να είναι στον σκληρό δίσκο και όχι στην μνήμη

- **Η λύση είναι να εισάγουμε έναν επιπλέον μηχανισμό cache για να αποθηκεύουμε τον πίνακα σελίδων**

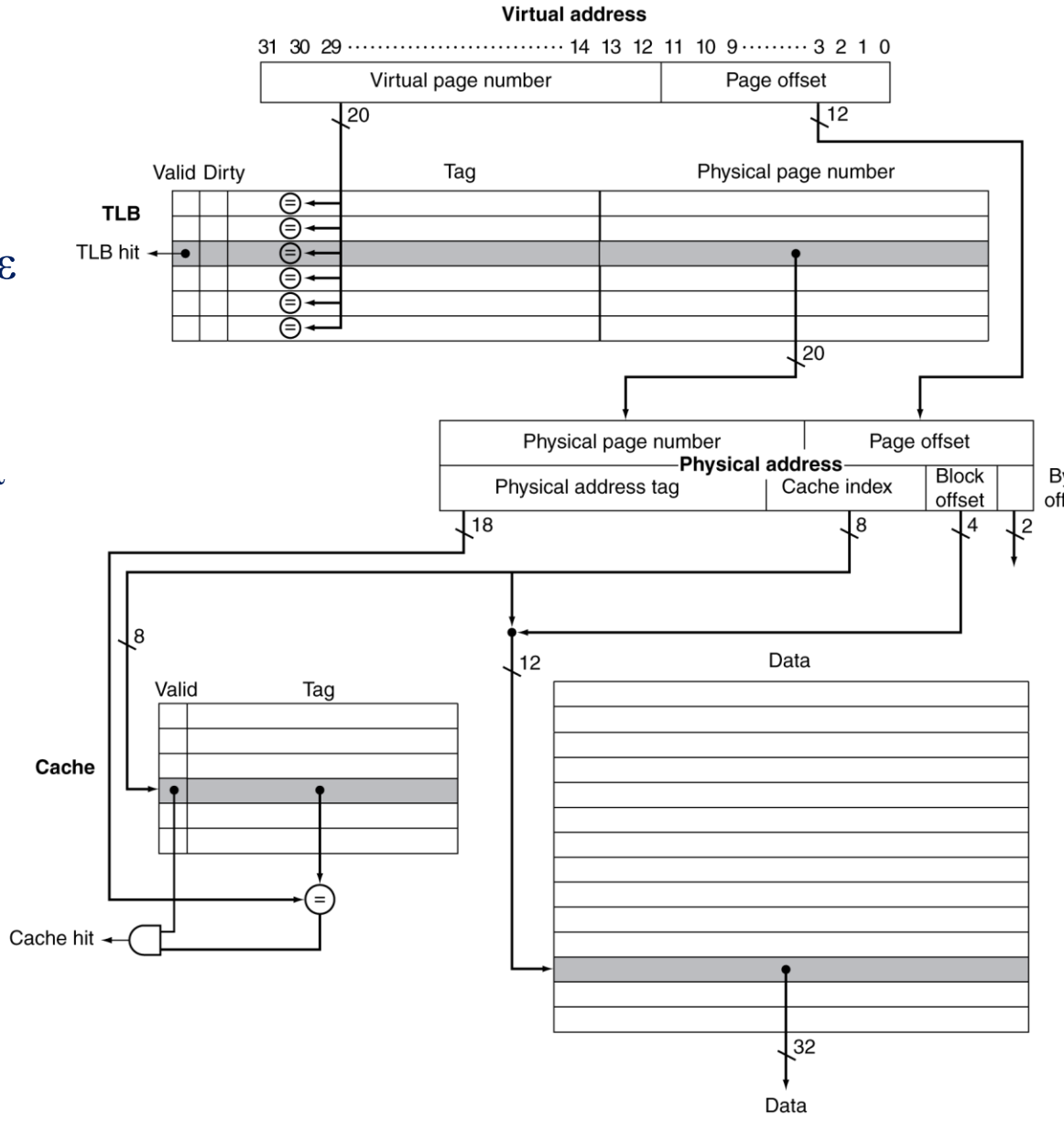
Κρυφή μνήμη αναζήτησης μετάφρασης (Translation-lookaside buffer, TLB)

- Το TLB είναι μια μνήμη cache που αποθηκεύει μέρος του πίνακα σελίδων
- Συνήθως fully-associative, με μικρό μέγεθος 16-512 καταχωρήσεις, μέγεθος block 4-8 bytes και Miss rate πολύ μικρό 0,01-1%
- Είναι χρονικά η πρώτη μνήμη που προσπελαύνει ο επεξεργαστής για εντολές και δεδομένα
- Ξεχωριστά Instruction TLB (ITLB) και Data TLB (DTLB)



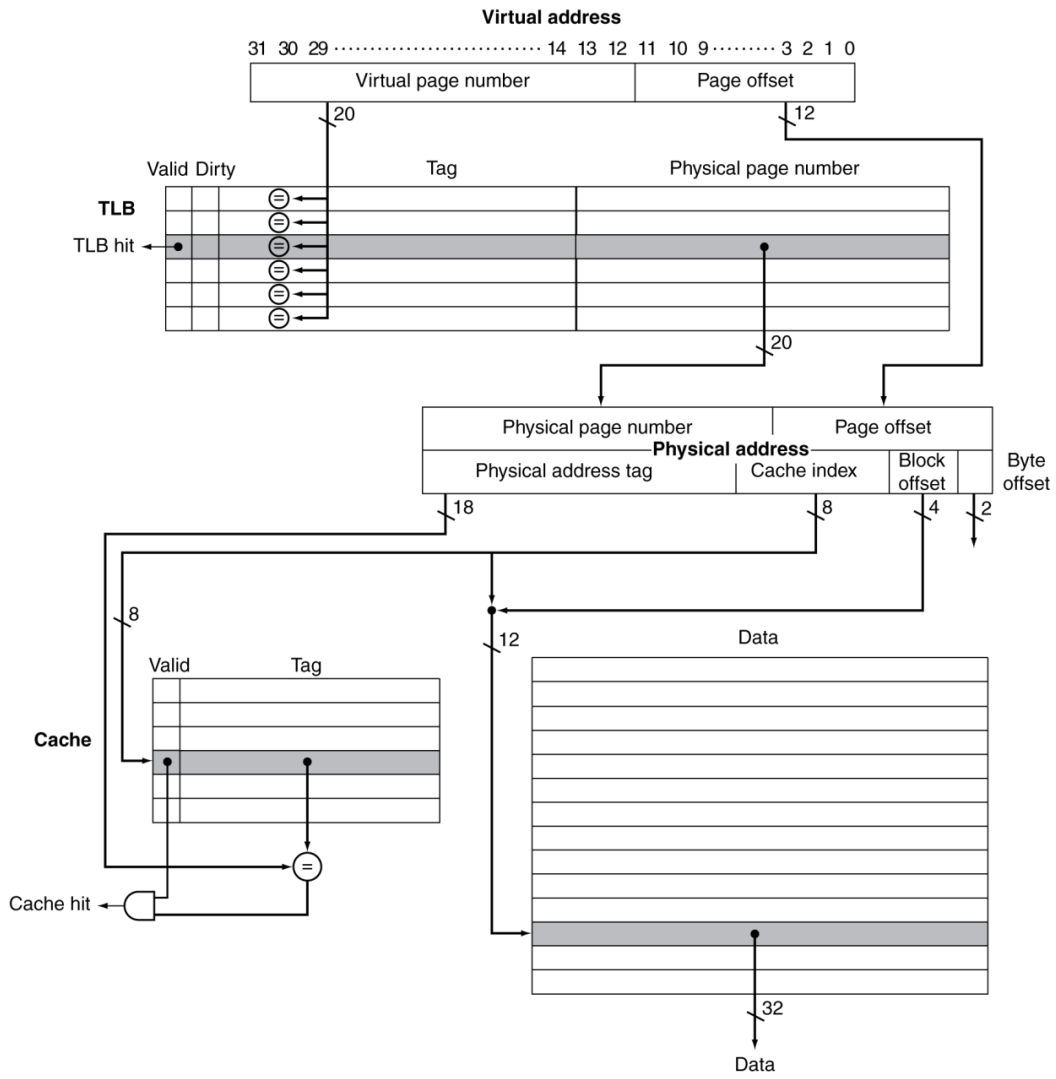
Σχέση TLB και Caches

- Σε περίπτωση **TLB Hit** η διεύθυνση της φυσικής σελίδας στο TLB μαζί με την μετατόπιση σελίδας (page offset) αποτελεί την φυσική διεύθυνση.
- Η φυσική διεύθυνση χρησιμοποιείται για την προσπέλαση της cache.
- Οι μνήμες cache (όλων των επιπέδων) είναι virtually -indexed, physically-addressed.



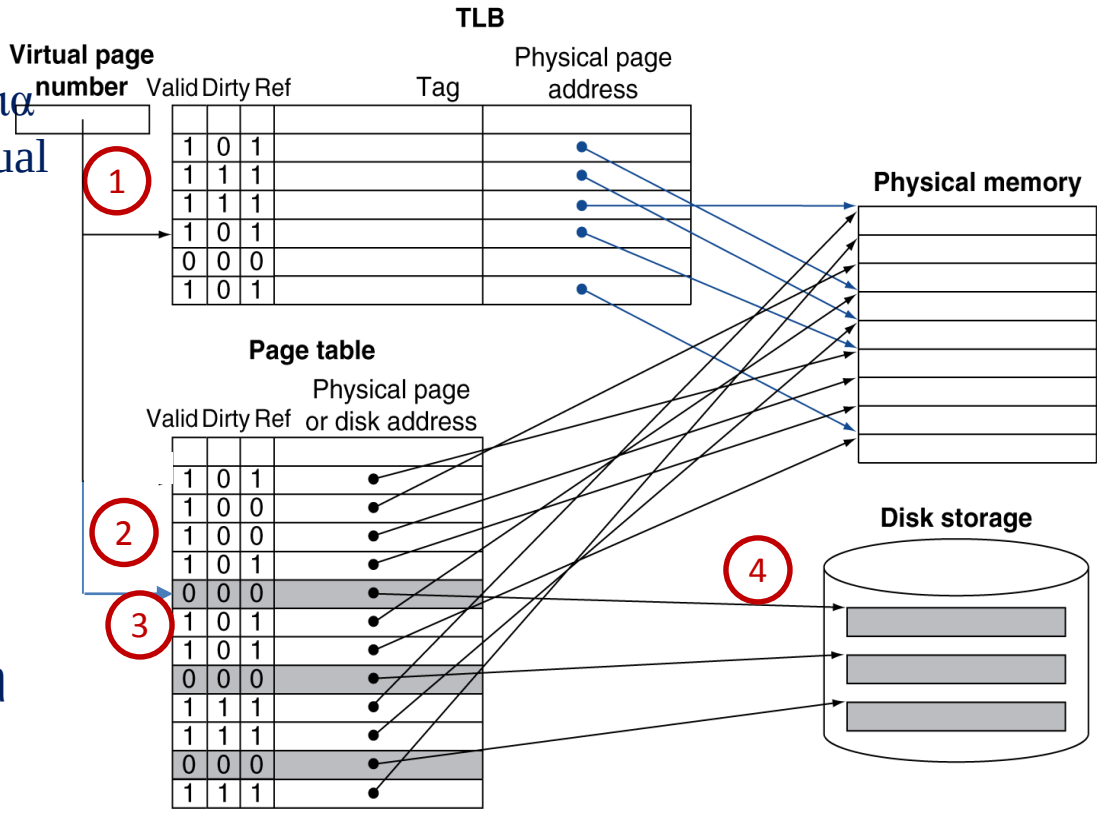
Σχέση TLB και Caches

- Σε περίπτωση **TLB Miss**, η καταχώρηση πρέπει να έρθει από τον πίνακα σελίδων (page table) στο TLB. Δηλ. πρέπει να κάνουμε προσπέλαση στην κύρια μνήμη.
- Το λειτουργικό σύστημα το κάνει αυτό.
- Η χειρότερη περίπτωση είναι η φυσική σελίδα που ζητάμε να μην είναι στην κύρια μνήμη, αλλά να πρέπει να την διαβάσουμε από τον σκληρό δίσκο. Αυτό είναι το σφάλμα σελίδας (page fault).
- Το λειτουργικό σύστημα διακόπτει την εκτέλεση του προγράμματος μας και αναλαμβάνει να διαβάσει την σελίδα από τον σκληρό δίσκο.



Σφάλμα Σελίδας (Page Fault)

1) Ο επεξεργαστής στέλνει την VA (Virtual Address) στο TLB.
   - Συγκεκριμένα στέλνει μόνο το Virtual Page Number (VPN)
2) TLB miss. Πήγαινε στο Page Table στην κύρια μνήμη να διαβάσεις την μετάφραση από Virtual σε Physical page number
3) Το Valid bit είναι 0. Άρα Page Table Miss → Page Fault
4) Το Λειτουργικό Σύστημα διακόπτει την διεργασία του χρήστη, και αναλαμβάνει να φέρει την σελίδα (και ίσως και άλλες πολλές μαζί, prefetching) στην κύρια μνήμη.
5) Όσο συμβαίνει αυτή η μεταφορά, το Λειτουργικό Σύστημα δίνει την CPU σε άλλη διεργασία.
6) Όταν έρθει η σελίδα (faulty page), ανανεώνουμε το page table για την αρχική διεργασία.
7) Όταν έρθει η σειρά της, η αρχική διεργασία ξεκινάει πάλι από την εντολή που δημιούργησε το Page fault.

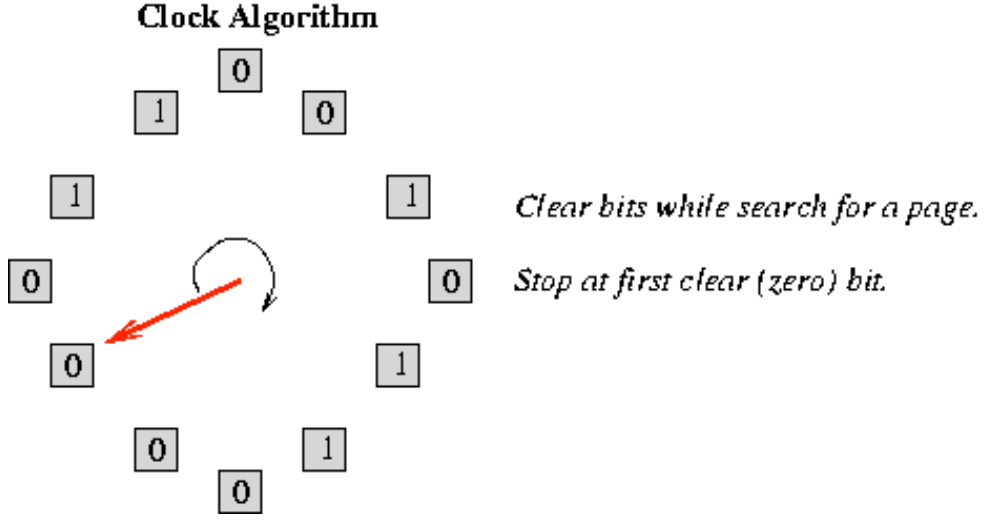


Αλγόριθμοι αντικατάστασης φυσικής σελίδας

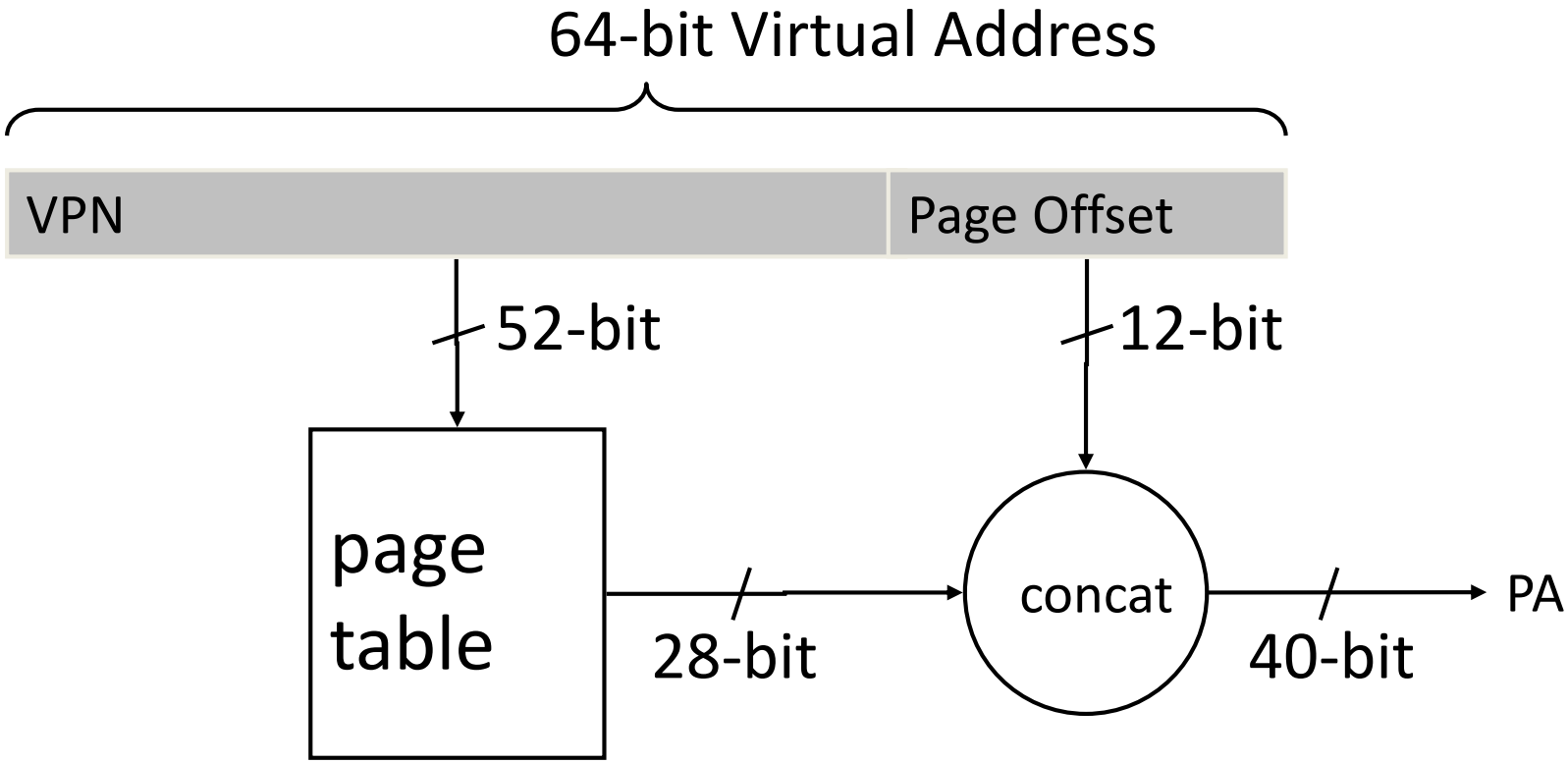
- Τι θα συμβεί εάν η φυσική μνήμη είναι γεμάτη και δεν υπάρχει περιοχή για να φέρουμε μιά καινούργια φυσική σελίδα μετά από ένα σφάλμα σελίδας;
  - Ποιά φυσική σελίδα θα πρέπει να αντικατασταθεί απο αυτές που είναι ήδη στην φυσική μνήμη;
  - Πρόβλημα ανάλογο με την αντικατάσταση ενός cache block
- Μπορούμε να χρησιμοποιήσουμε LRU (Least Recently Used);
  - Με 8GB μνήμη και 4KB μέγεθος σελίδας, έχουμε 2 M σελίδες. Δύσκολο να κρατήσουμε counters για τόσες σελίδες.
- Συνήθως χρησιμοποιούνται προσεγγιστικοί αλγόριθμοι του LRU
  - Πχ. CLOCK αλγόριθμος

Αλγόριθμος CLOCK για αντικατάσταση φυσικής σελίδας

- Κρατάμε μιά κυκλική λίστα από τις φυσικές σελίδες της μνήμης
- Επίσης έναν pointer (hand) στην τρέχουσα σελίδα της λίστας
- Όταν μία φυσική σελίδα προσπελαύνεται (read/write), θέτουμε το Reference Bit (R) στο αντίστοιχο page table entry (PTE)
- Αντικαθιστούμε την πρώτη φυσική σελίδα στην μνήμη που συναντάμε όταν διασχίζουμε την λίστα ξεκινώντας από την τελευταία θέση του pointer και που έχει R=0
- Κατά την διάρκεια του αλγορίθμου, θέτουμε R=1 σε όλες τις σελίδες που συναντούμε



Άλλο πρόβλημα: Μέγεθος του Page Table



Έστω 64-bit VA και 40-bit PA, με 4KB μέγεθος σελίδας.

Πόσο μεγάλο είναι το page table?

2^{52} εισόδους * ~4 bytes ≈ 2^{54} bytes!

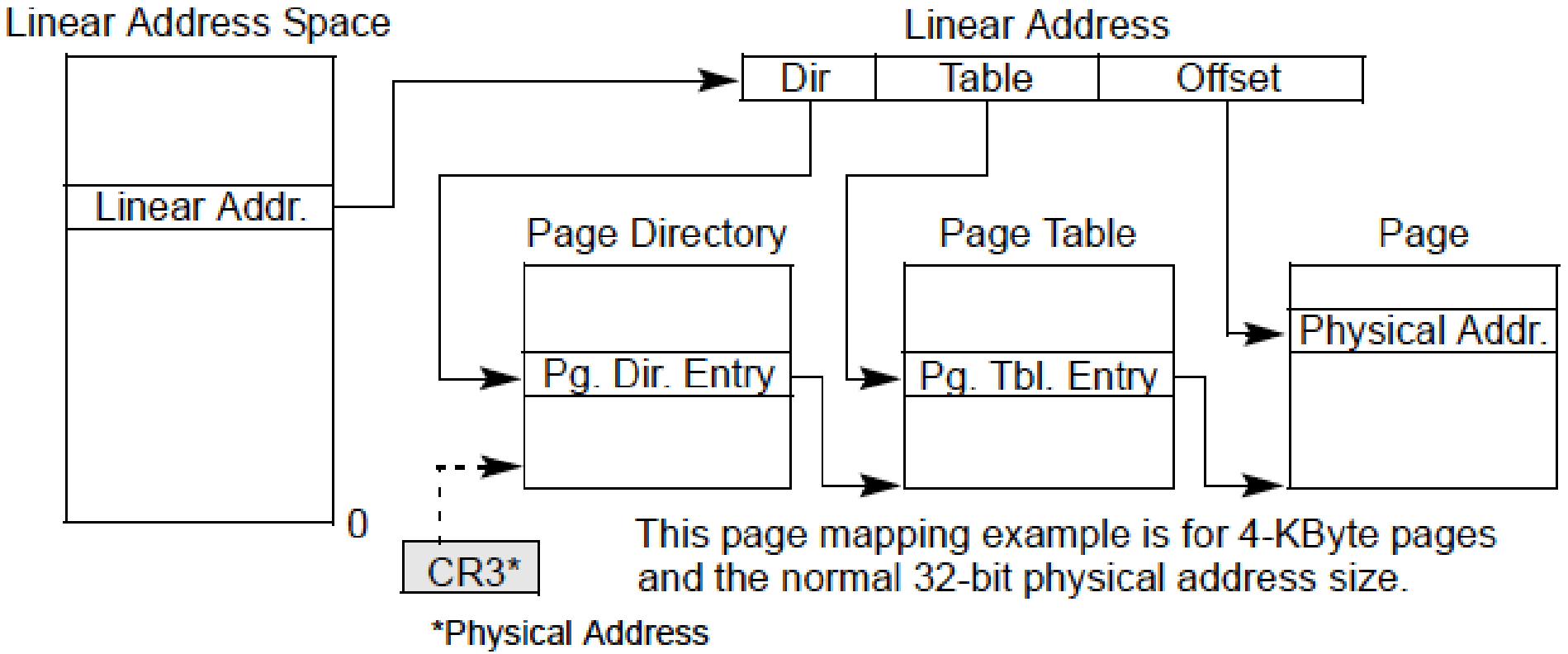
και αυτό μόνο για μία διεργασία

όσο μικρή και να είναι

Λύση: Πίνακας σελίδας πολλαπλών επιπέδων

Αρχιτεκτονική Intel x86*

*Η Intel χρησιμοποιεί τους όρους linear address (αντί virtual address) και real address (αντί physical address)



Σε κάθε context switch, το λειτουργικό σύστημα κάνει switch και φέρνει το νέο CR3 στον καταχωρητή.

Προστασία μνήμης (Memory Protection)

- Μέσω της έμμεσης προσπέλασης, η εικονική μνήμη εμποδίζει μια διεργασία να διαβάσει και να γράψει το κομμάτι της φυσικής μνήμη μιας άλλης διεργασίας.
- Για να το εξασφαλίσουμε αυτό θα πρέπει να εμποδίσουμε μία διεργασία (user process) να μπορεί να αλλάξει τα περιεχόμενα του πίνακα σελίδων.
- Αυτό επιτυγχάνεται ως εξής:
  - Ο επεξεργαστής λειτουργεί σε δύο καταστάσεις (modes): user & kernel mode.
    - Το λειτουργικό σύστημα τρέχει πάντα όταν η CPU είναι σε kernel mode.
  - Η κατάσταση του πίνακα σελίδων μπορεί να αλλάξει μόνο από μία διεργασία που τρέχει σε kernel mode. Τι μπορεί να αλλάξει σε kernel mode?
    - Πίνακα σελίδων (Page Table)
    - Καταχωρητής πίνακα σελίδων (Page Table Base Pointer)
    - Το TLB

Προστασία μνήμης (Memory Protection)

- Η πληροφορία για το εάν μια σελίδα μπορεί να προσπελασθεί από μία διεργασία περιέχεται μέσα στο page table της διεργασίας
- Αποθηκεύουμε επιπλέον control bits σε κάθε είσοδο στο page table που δείχνουν το επίπεδο προστασίας κάθε σελίδας
- Read/Write/Execute
- Η επιβολή του ελέγχου προστασίας (memory protection) από το σύστημα γίνεται κατά την διάρκεια της μετάφρασης της εικονικής μνήμης

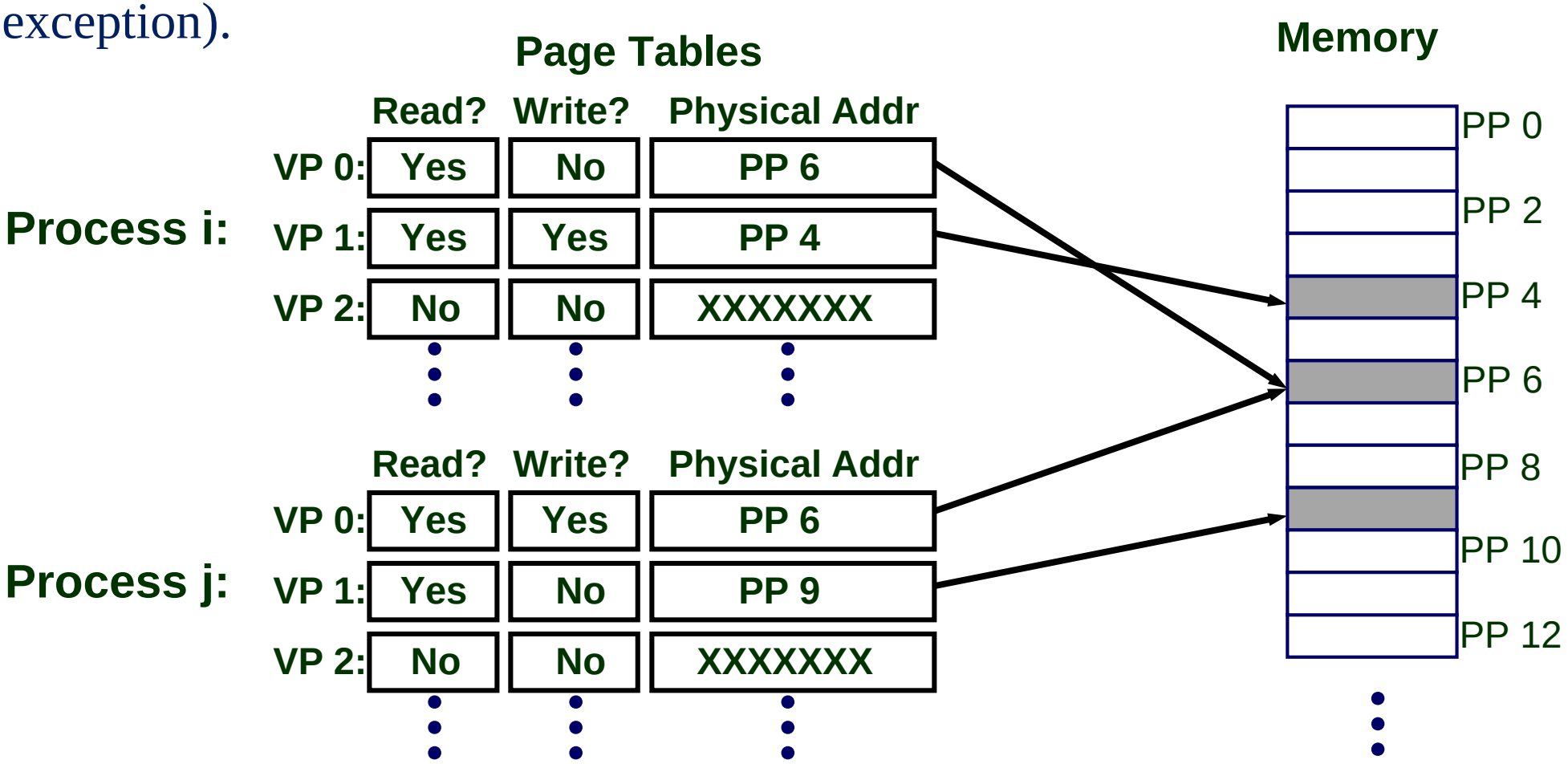
→ Τα συστήματα εικονικής μνήμης έχουν δύο βασικούς ρόλους:

1. Μετάφραση από εικονική σε φυσική μνήμη
2. Έλεγχος προσπέλασης μνήμης (memory protection)

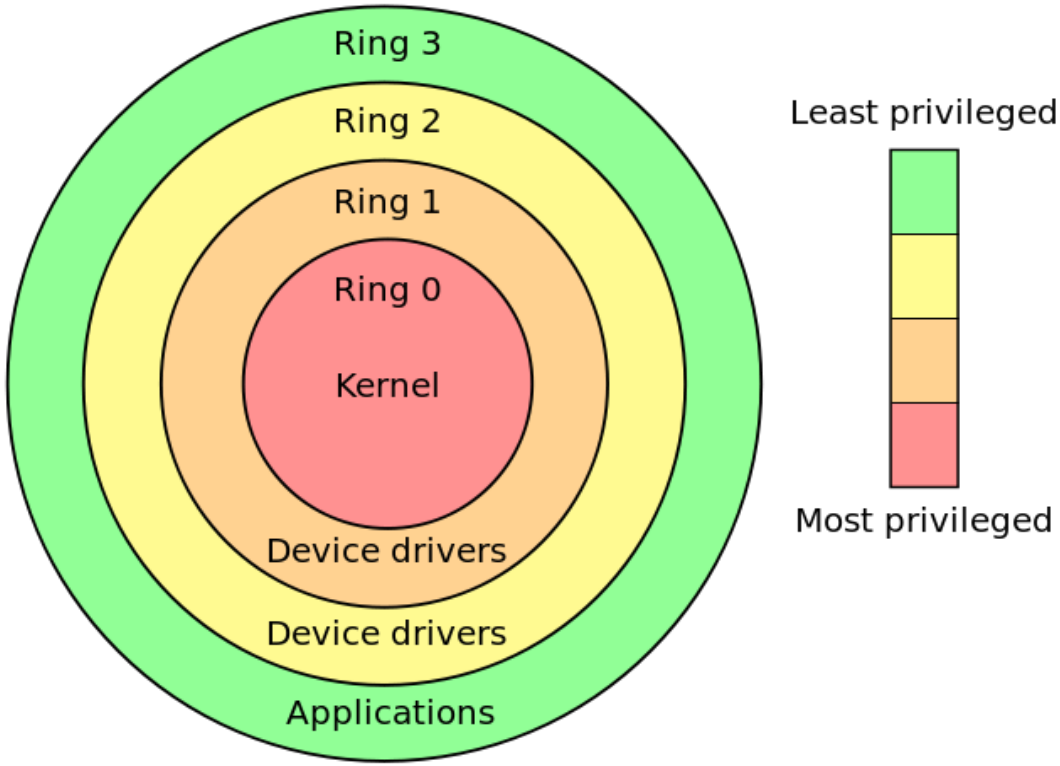
Προστασία μνήμης (Memory Protection)

Extra permission bits που ελέγχονται σε **κάθε** προσπέλαση στην μνήμη

Δημιουργείται exception και διακοπή προγράμματος εάν υπάρξει παράβαση από κάποια προσπέλαση στην μνήμη (Access Protection exception).



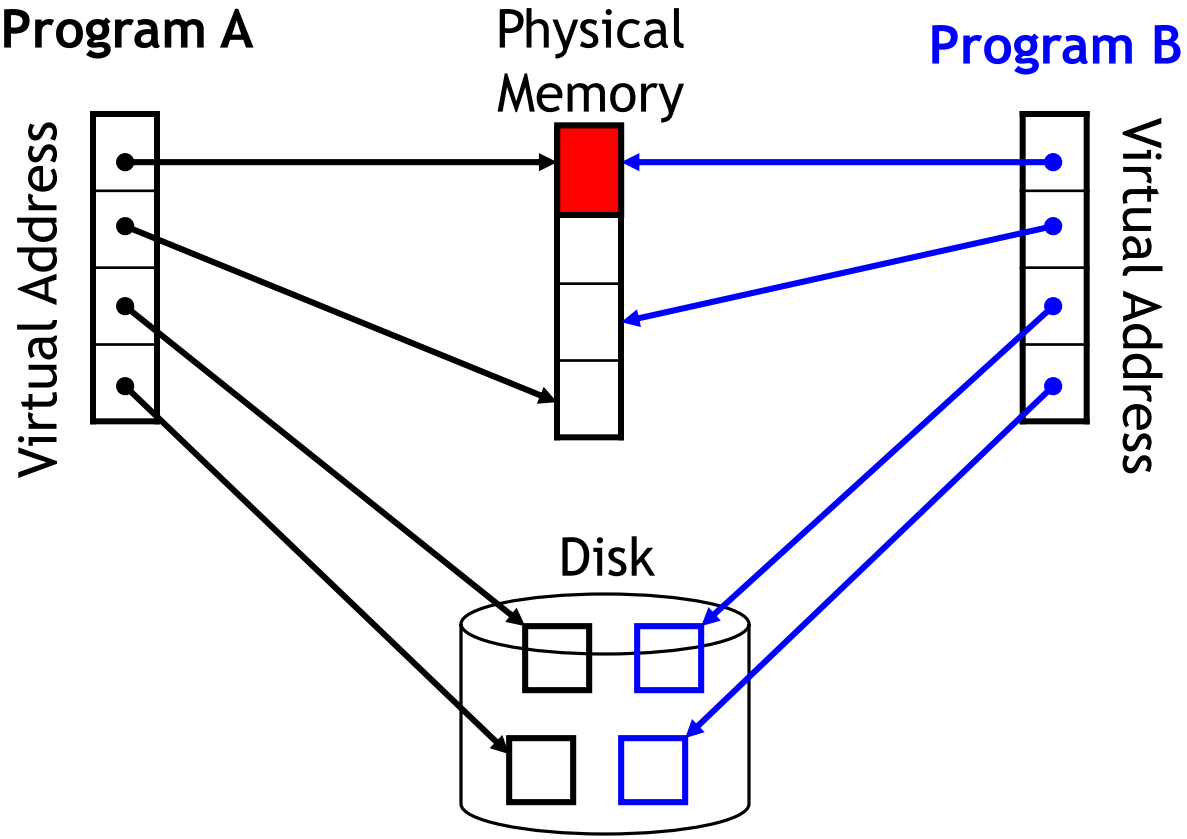
Επίπεδα Προστασίας στον x86



Πηγή: *https://en.wikipedia.org/wiki/Protection_ring*

Memory Sharing

- Δύο διεργασίες μπορούν να μοιράζονται την ίδια φυσική μνήμη με το να επιτρέπουν τους πίνακες σελίδων τους να δείχνουν στις ίδιες φυσικές διευθύνσεις
- Για παράδειγμα, εάν τρέχετε δύο αντίγραφα του ίδιου προγράμματος, το λειτουργικό σύστημα θα επιτρέψει την ταυτόχρονη προσπέλαση της ίδιας φυσικής μνήμης.



Prefetching

- Πολλά συστήματα υψηλής απόδοσης κάνουν aggressive data and instruction prefetching.
  - Πρός την L3 cache συνήθως
- Το prefetching διακόπτεται αμέσως όταν προκαλέσει ένα TLB miss.
  - Δεν θέλουμε το prefetch να είναι υπεύθυνο για τόσο μεγάλη διακοπή της ροής του προγράμματος.
- Οι HW prefetchers φέρνουν δεδομένα μέσα σε μία σελίδα (page).
  - Συνήθως δουλεύουν με φυσικές μνήμες και δεν χρησιμοποιούν το TLB.

Μνήμες σε πραγματικό σύστημα Core-i7 ιεραρχία μνήμης