Εικονική Μνήμη (virtual memory)

- Πολλά προγράμματα εκτελούνται ταυτόχρονα σε ένα υπολογιστή
- Η συνολική μνήμη που απαιτείται είναι μεγαλύτερη από το μέγεθος της RAM
- Αρχή τοπικότητας (η μνήμη χρησιμοποιείται από τα ενεργά τμήματα των προγραμμάτων – working sets)
- Πρέπει να εξασφαλίσουμε προστασία μεταξύ των προγραμμάτων

Εικονική Μνήμη (virtual memory)

- Την ώρα του compile, δεν γνωρίζουμε ποια προγράμματα θα μοιράζονται τη μνήμη
  - Δυναμική εκτέλεση προγραμμάτων
- Θέλουμε κάθε πρόγραμμα να νομίζει ότι έχει τη δική του μνήμη
- Θέλουμε κάθε πρόγραμμα να νομίζει ότι έχει απεριόριστη μνήμη (μεγαλύτερη από τη RAM)

Εικονική Μνήμη (virtual memory)

- Η Virtual memory ελέγχει 2 επίπεδα της ιεραρχίας μνήμης:
  - Κύρια μνήμη (DRAM)
  - Μαζική αποθήκευση (συνήθως μαγνητικοί δίσκοι)
- Η κύρια μνήμη διαιρείται σε blocks κατανεμημένες σε διαφορετικές τρέχουσες διεργασίες του συστήματος:
  - Blocks καθορισμένου μεγέθους: Pages (μέγεθος 4k έως 64k bytes).
  - Blocks μεταβλητού μεγέθους : Segments (μέγεθος το πολύ 216 μέχρι 232)
- Σε δεδομένο χρόνο, για κάθε τρέχουσα διεργασία, ένα κομμάτι των δεδομένων ή του κώδικα φορτώνεται στην κύρια μνήμη ενώ το υπόλοιπο είναι διαθέσιμο μόνο στους μαγνητικούς δίσκους.
- Ένα block κώδικα ή δεδομένων που χρειάζεται για την εκτέλεση μιας διεργασίας αλλά δεν υπάρχει στη κύρια μνήμη έχει ως αποτέλεσμα ένα page fault (address fault) και το block πρέπει να φορτωθεί στην κύρια μνήμη από το δίσκο ή τον χειριστή του λειτουργικού συστήματος (OS handler).
- Ένα πρόγραμμα μπορεί να εκτελεστεί σε οποιαδήποτε θέση της κύριας μνήμης ή του δίσκου χρησιμοποιώντας έναν μηχανισμό επανατοποθέτησης ο οποίος να ελέγχεται από το λειτουργικό σύστημα που να αντιστοιχεί τις διευθύνσεις από τον χώρο των virtual addresses (logical program address) στο χώρο των physical addresses (κύρια μνήμη, δίσκος).

Εικονική Μνήμη (virtual memory)

- Εικονική Μνήμη υλοποιεί μετάφραση του χώρου διευθύνσεων ενός προγράμματος σε φυσικές διευθύνσεις
- Διαδικασία μετάφρασης ενισχύει την προστασία

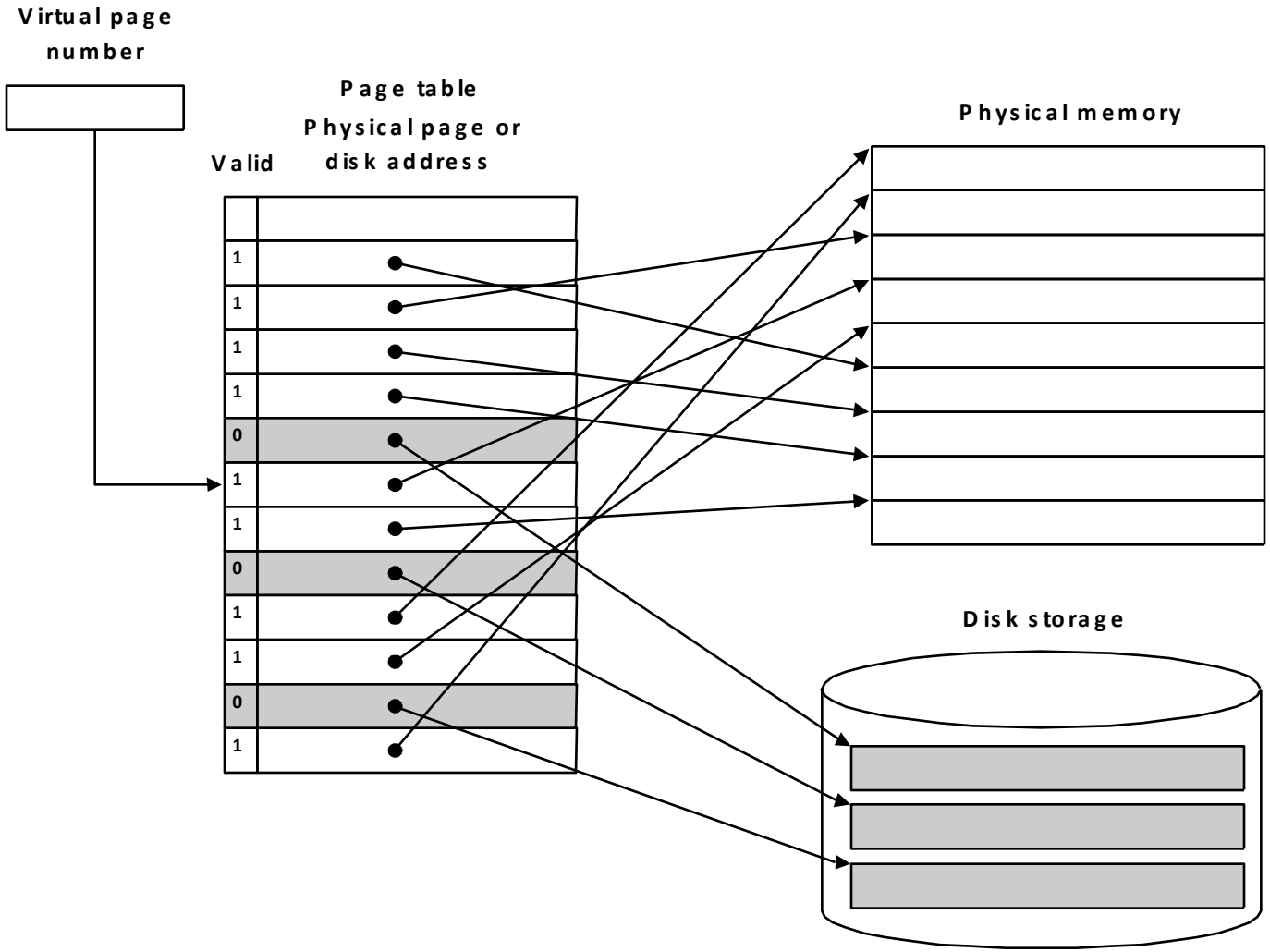
  (Εικονική μνήμη μοιάζει με την cache)

Τι βλέπει η εφαρμογή....

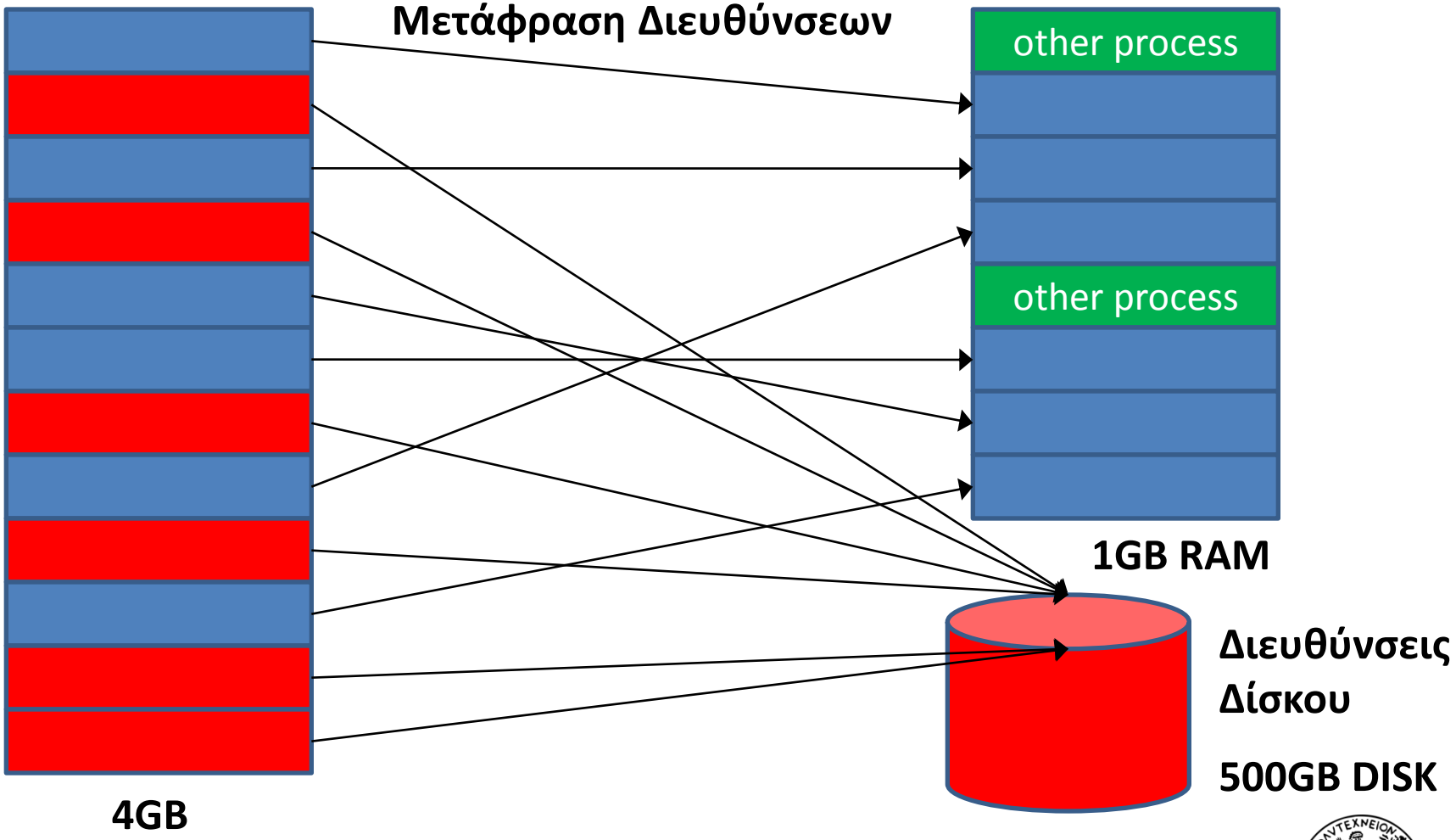
Εικονικές Διευθύνσεις

4GB

Μετάφραση των Virtual Addresses

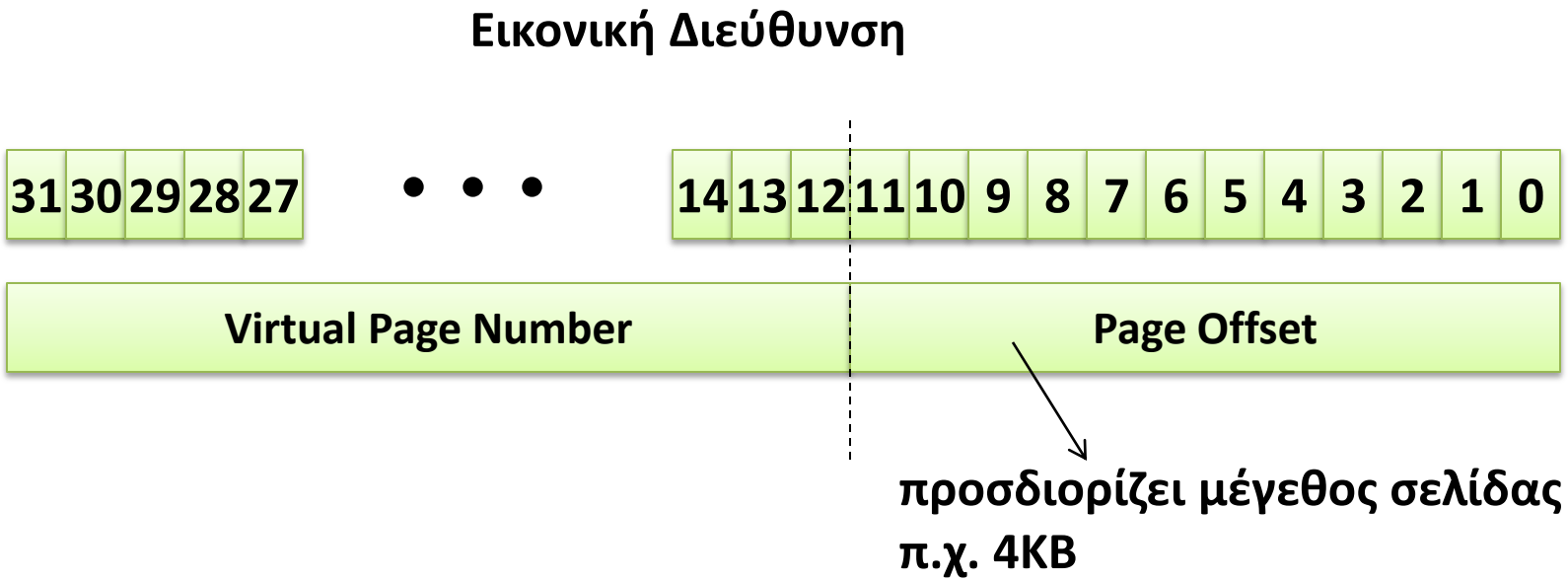


Τι συμβαίνει στην πραγματικότητα!!

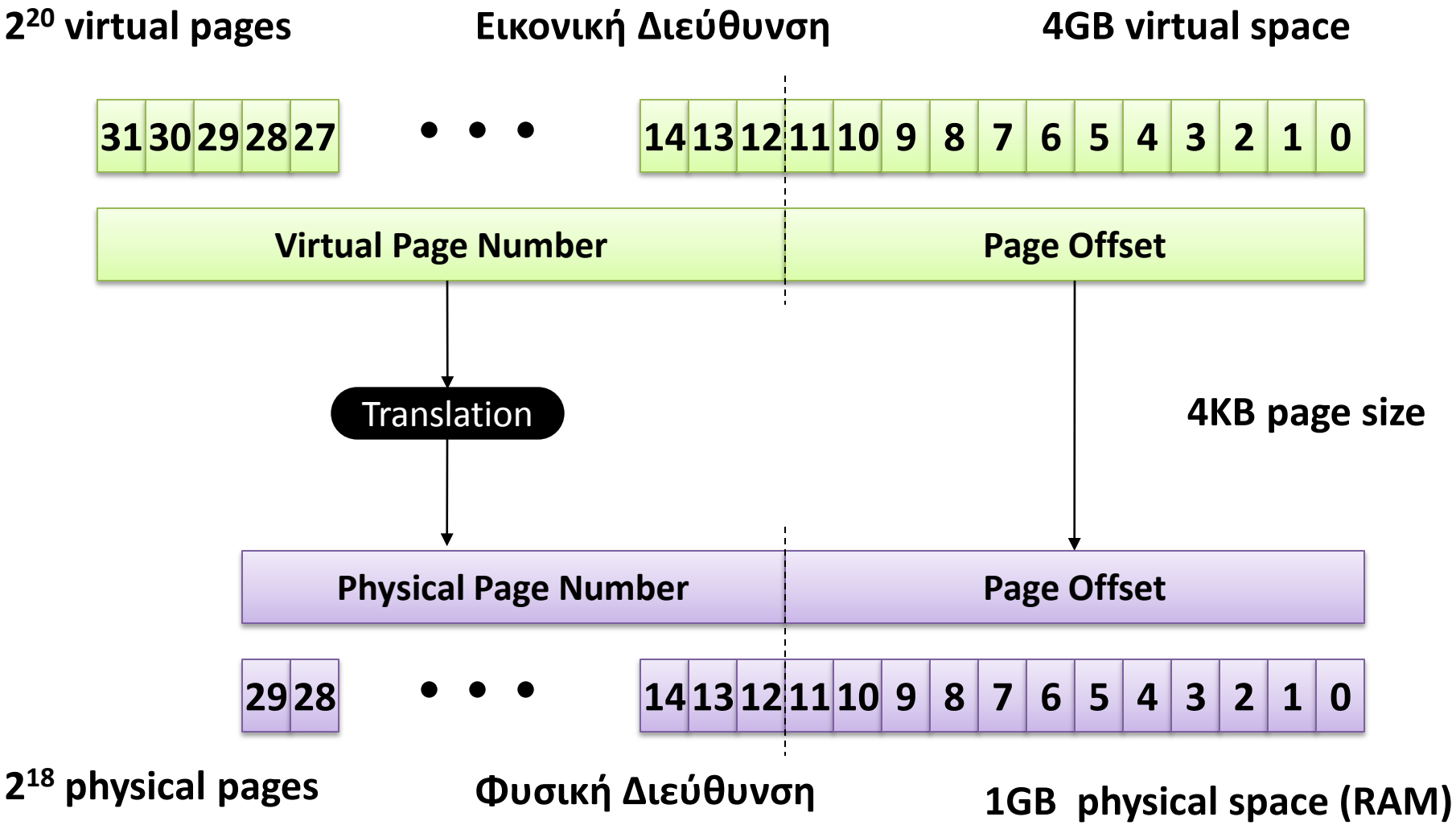


Εικονική Μνήμη

- Διεύθυνση χωρίζεται σε δύο τμήματα
  - αριθμός εικονικής σελίδας (virtual page number)
  - σχετική απόσταση σελίδας (page offset)

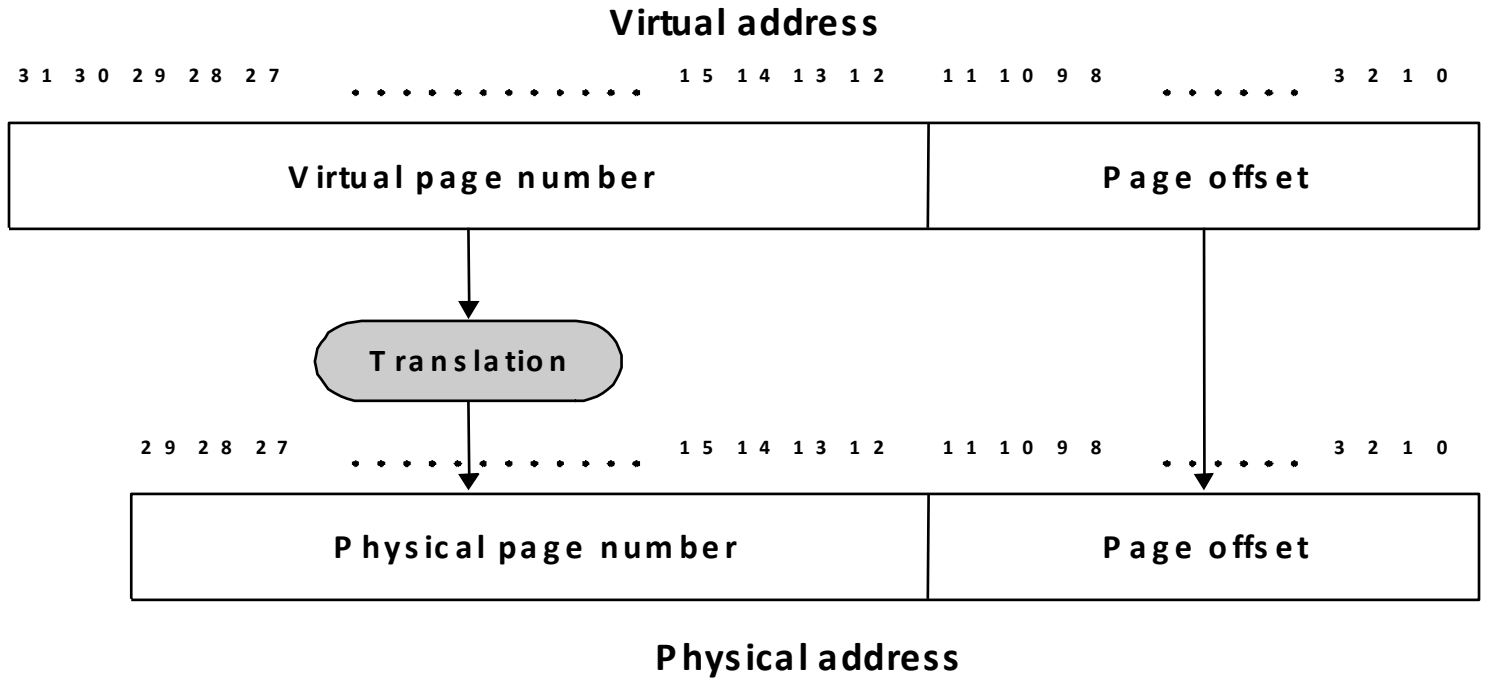


Μετάφραση Σελίδων



Πλεονεκτήματα

- Έχουμε την ψευδαίσθηση ότι διαθέτουμε περισσότερη φυσική κύρια μνήμη
- Επιτρέπει την επανατοποθέτηση των προγραμμάτων
- Προστατεύει από παράτυπη πρόσβαση στη μνήμη



Σχεδιαστικές Επιλογές

- Υψηλό κόστος αστοχίας – σφάλμα σελίδας (page fault)
  - Περιεχόμενα στο δίσκο (100.000 φορές πιο αργός)
- Σχεδιαστικές αποφάσεις για αποφυγή αστοχιών
  - «Μεγάλες» σελίδες (4KB – 16KB)
  - Fully associative τοποθέτηση σελίδων στη μνήμη
  - Σφάλματα αντιμετωπίζονται με λογισμικό – πιο έξυπνοι αλγόριθμοι επιλογής/τοποθέτησης σελίδων
  - Μόνο write-back (γιατί όχι write through;)

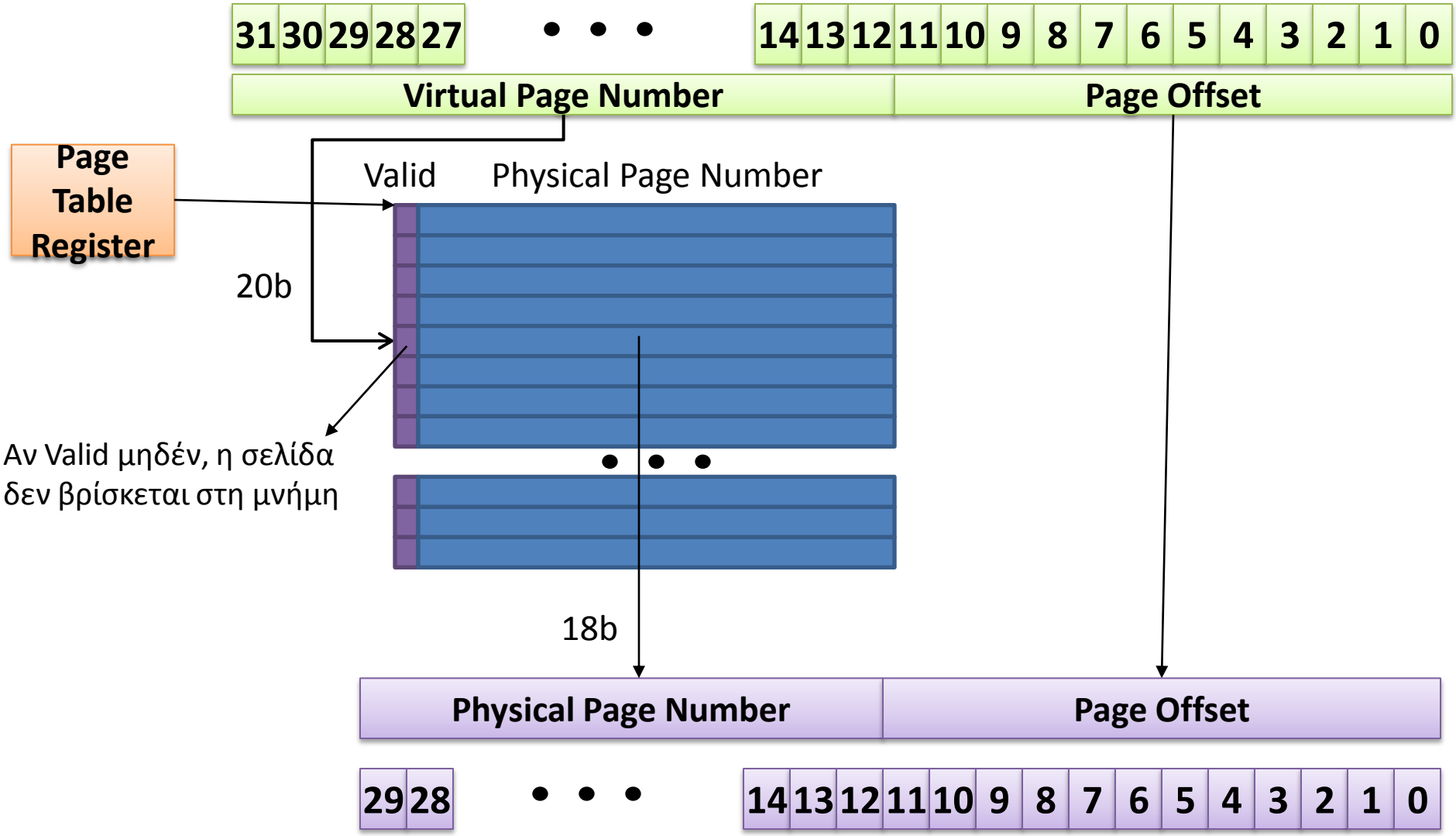
Σχεδιαστικά Λάθη

- Συνήθως virtual address είναι μεγαλύτερη από physical address
  - π.χ. 4GB virtual space vs 1GB physical space (RAM)
- Τι γίνεται όταν μέγεθος διεύθυνσης επεξεργαστή είναι μικρό σε σχέση με τις τεχνολογίες μνημών;
  - π.χ. μπορώ πολύ φτηνά να αγοράσω 8GB RAM, αλλά κυκλοφορούν μόνο 32bit υπολογιστές

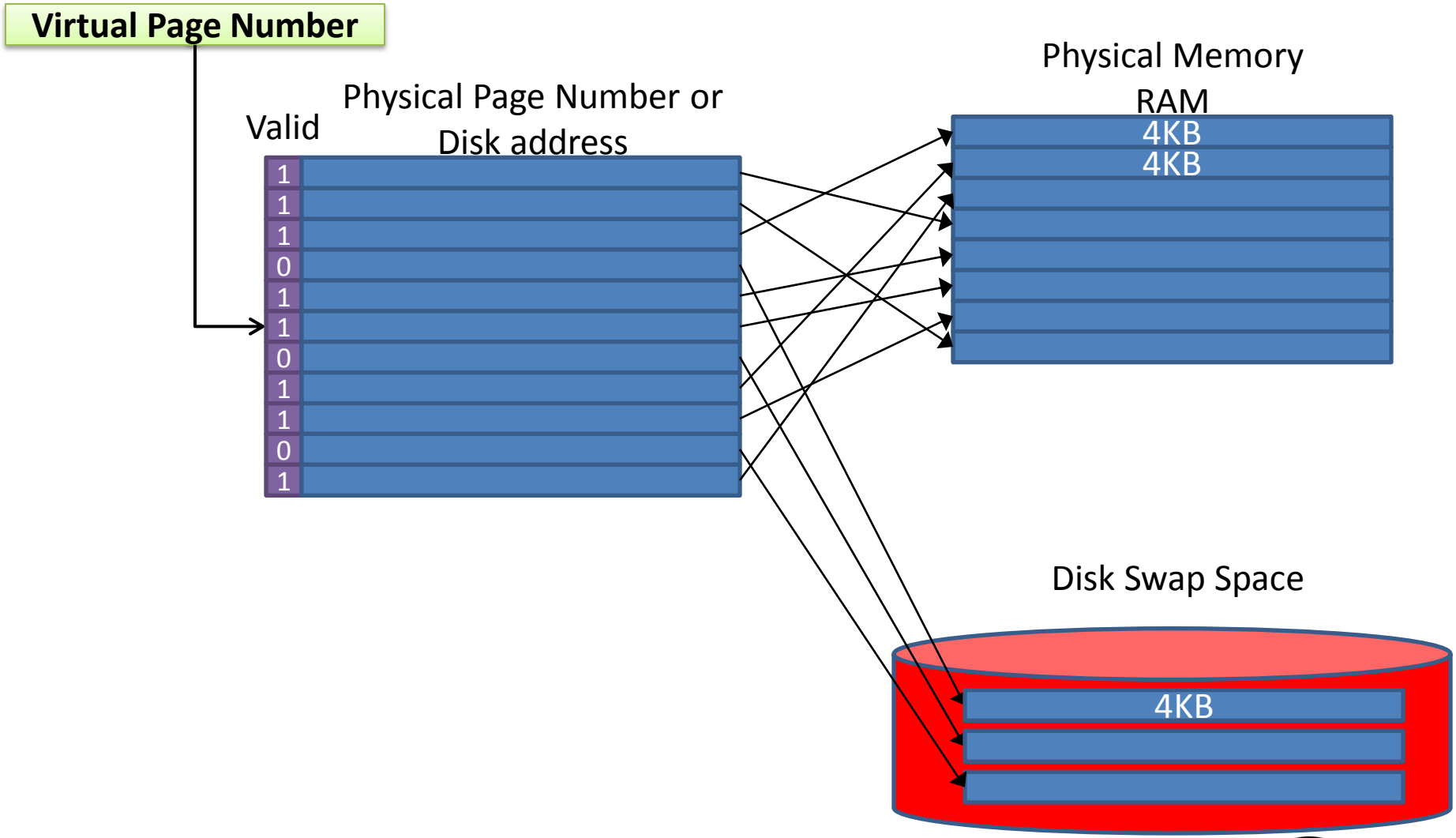
Σχεδιαστικά Λάθη

- Με 32bit υπολογιστή μπορούμε θεωρητικά να προσπελάσουμε 4GB και στην πράξη γύρω στα 3 – 3,5GB
- βλ. memory mapped devices
- βλ. OS kernel address space
- περισσότερα στα Λειτουργικά Συστήματα – ΡΟΗ Υ

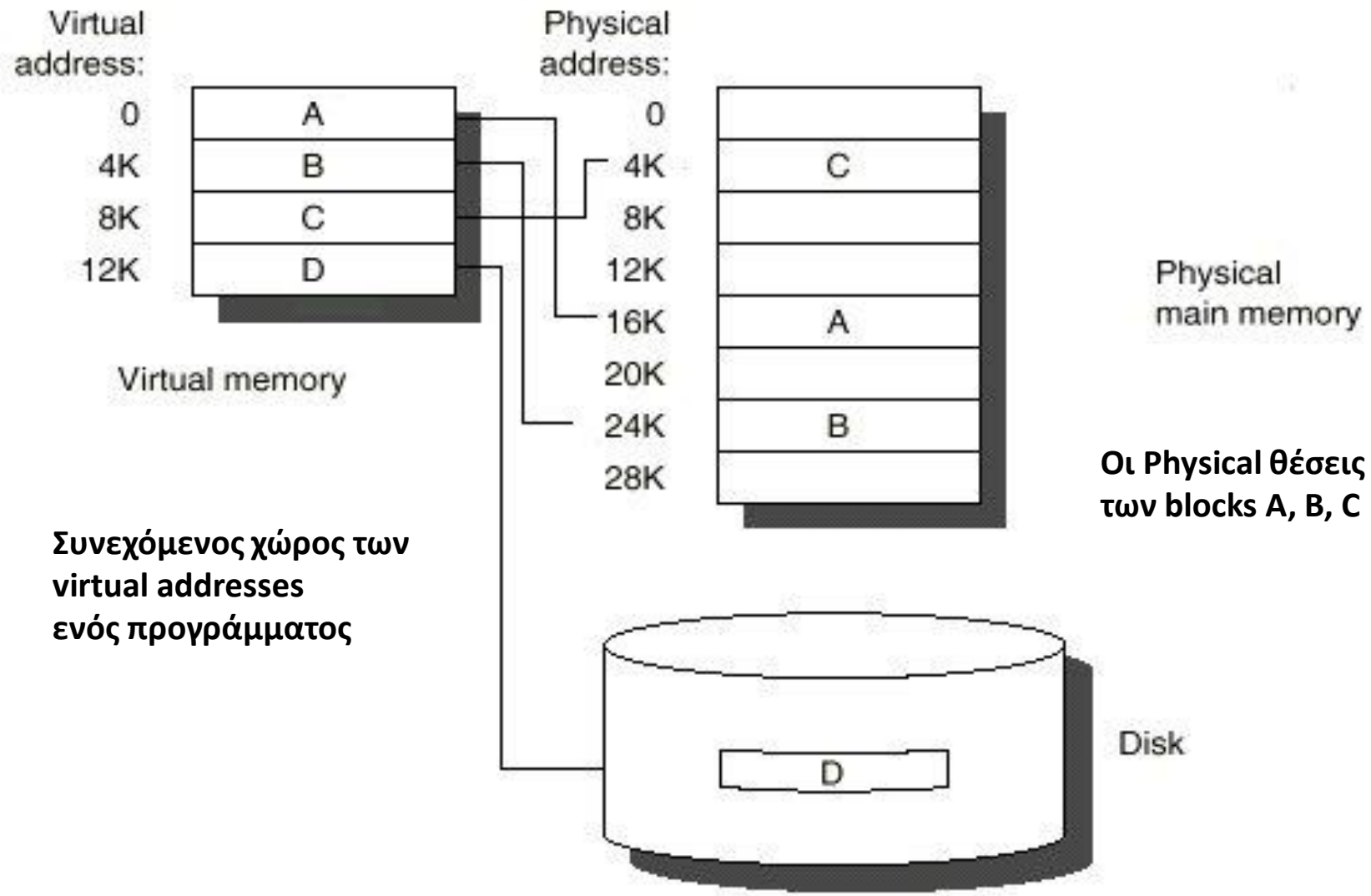
Μετάφραση Σελίδων – Πίνακας Σελίδων



Μετάφραση Σελίδων – Πίνακας Σελίδων



Μετάφραση Virtual -> Physical Addresses



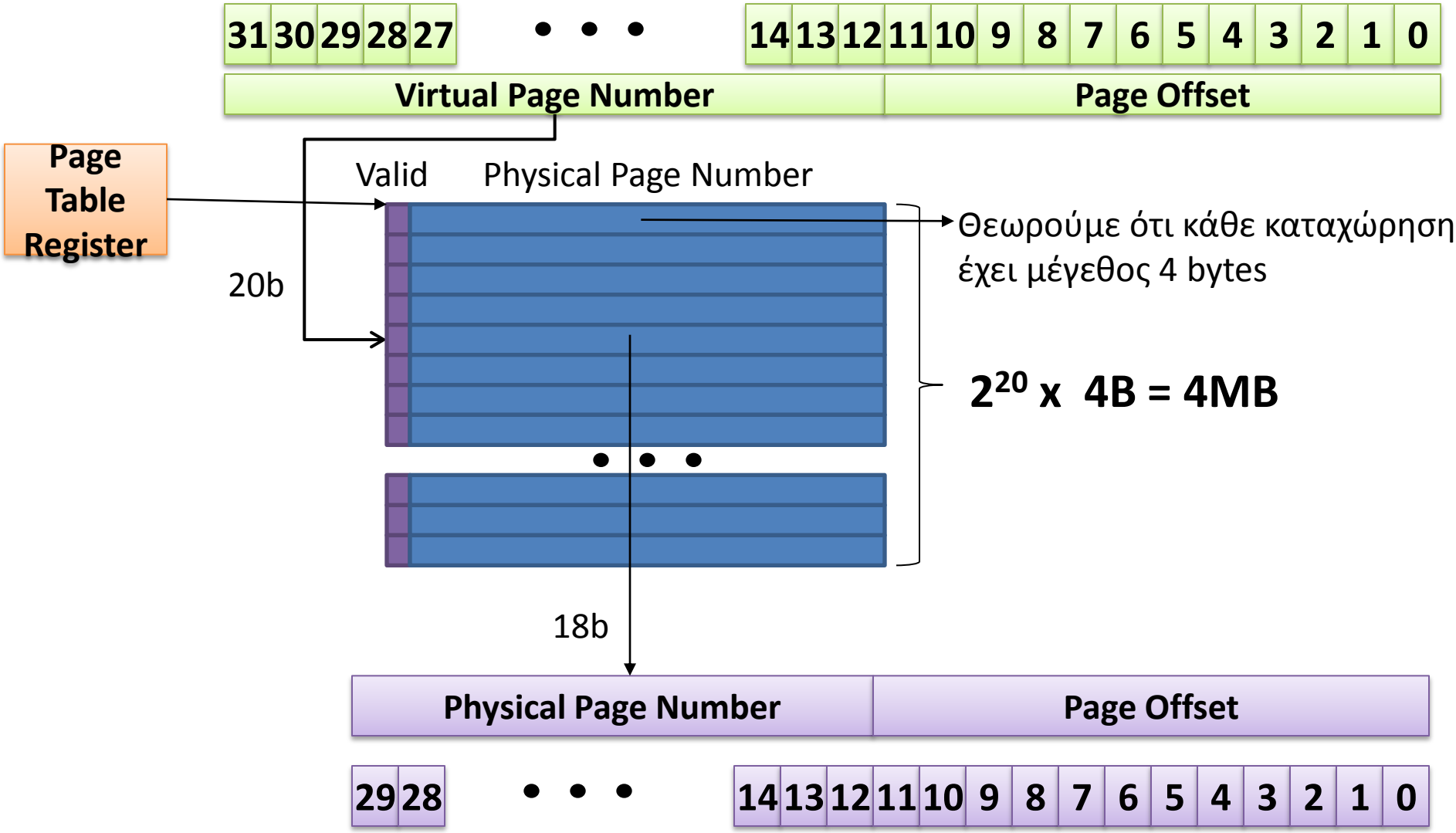
Συνεχόμενος χώρος των virtual addresses ενός προγράμματος

Οι Physical θέσεις των blocks A, B, C

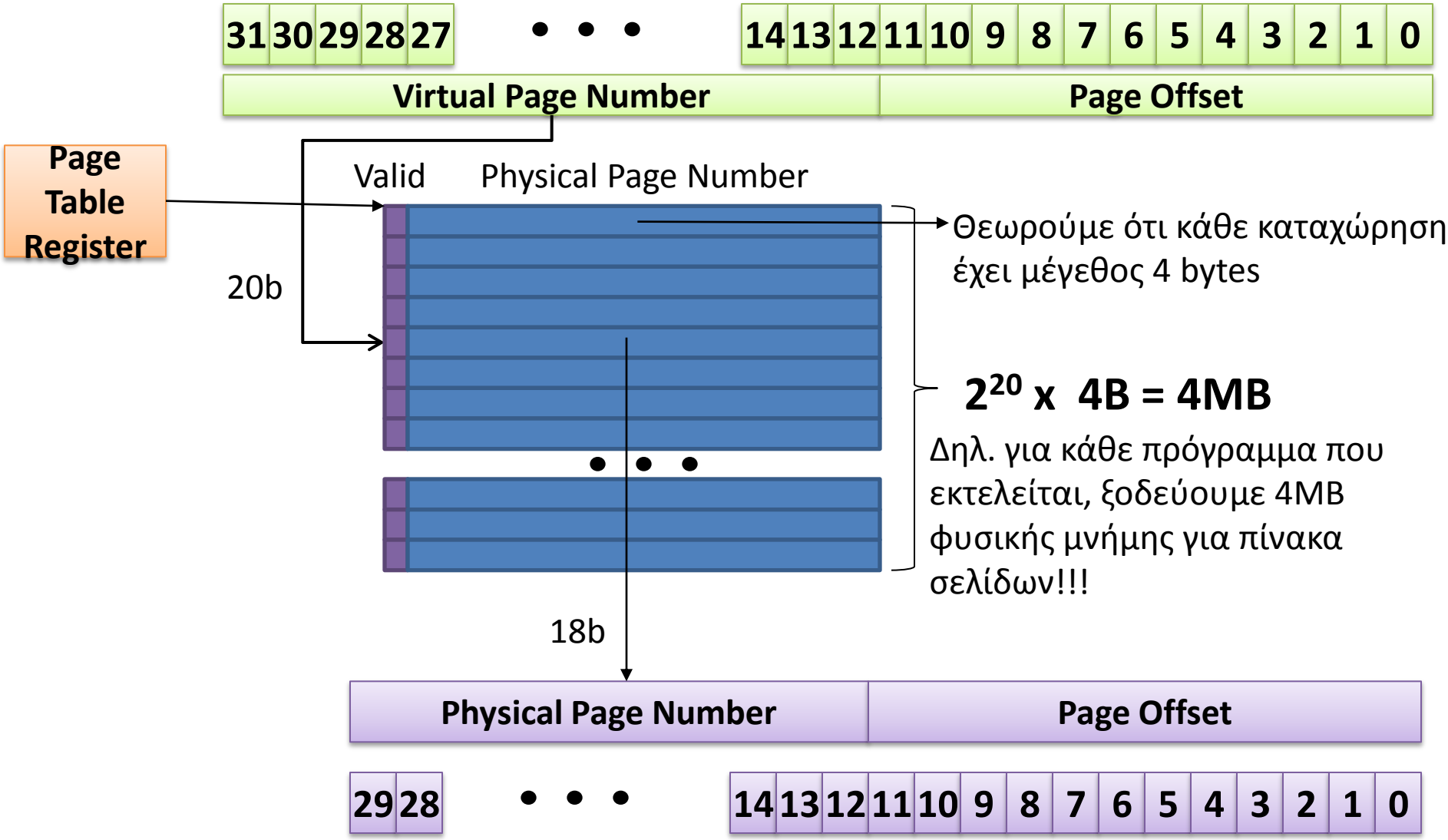
Εικονική Μνήμη

- Αν φυσική μνήμη γεμάτη (σύνολο προγραμμάτων μπορεί μεγαλύτερο από φυσική μνήμη), γίνεται αντικατάσταση
  - Σελίδα dirty γράφεται στο δίσκο (swap space)
  - Νέα σελίδα έρχεται στη φυσική μνήμη
  - Πρόγραμμα συνεχίζει εκτέλεση
- Αλγόριθμος αντικατάστασης LRU (μας θυμίζει τίποτα;) σε software-operating system

Μέγεθος Πίνακα Σελίδων



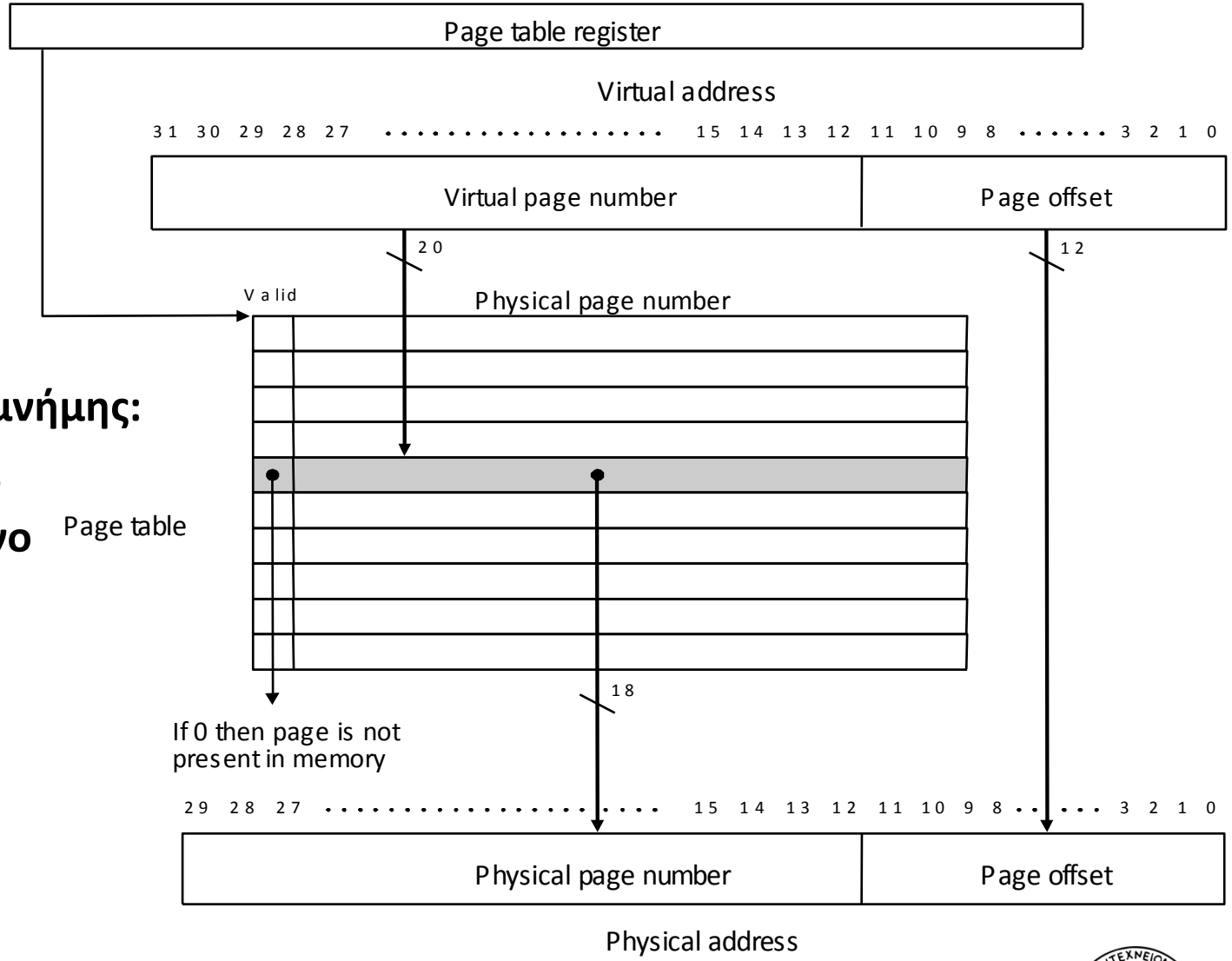
Μέγεθος Πίνακα Σελίδων



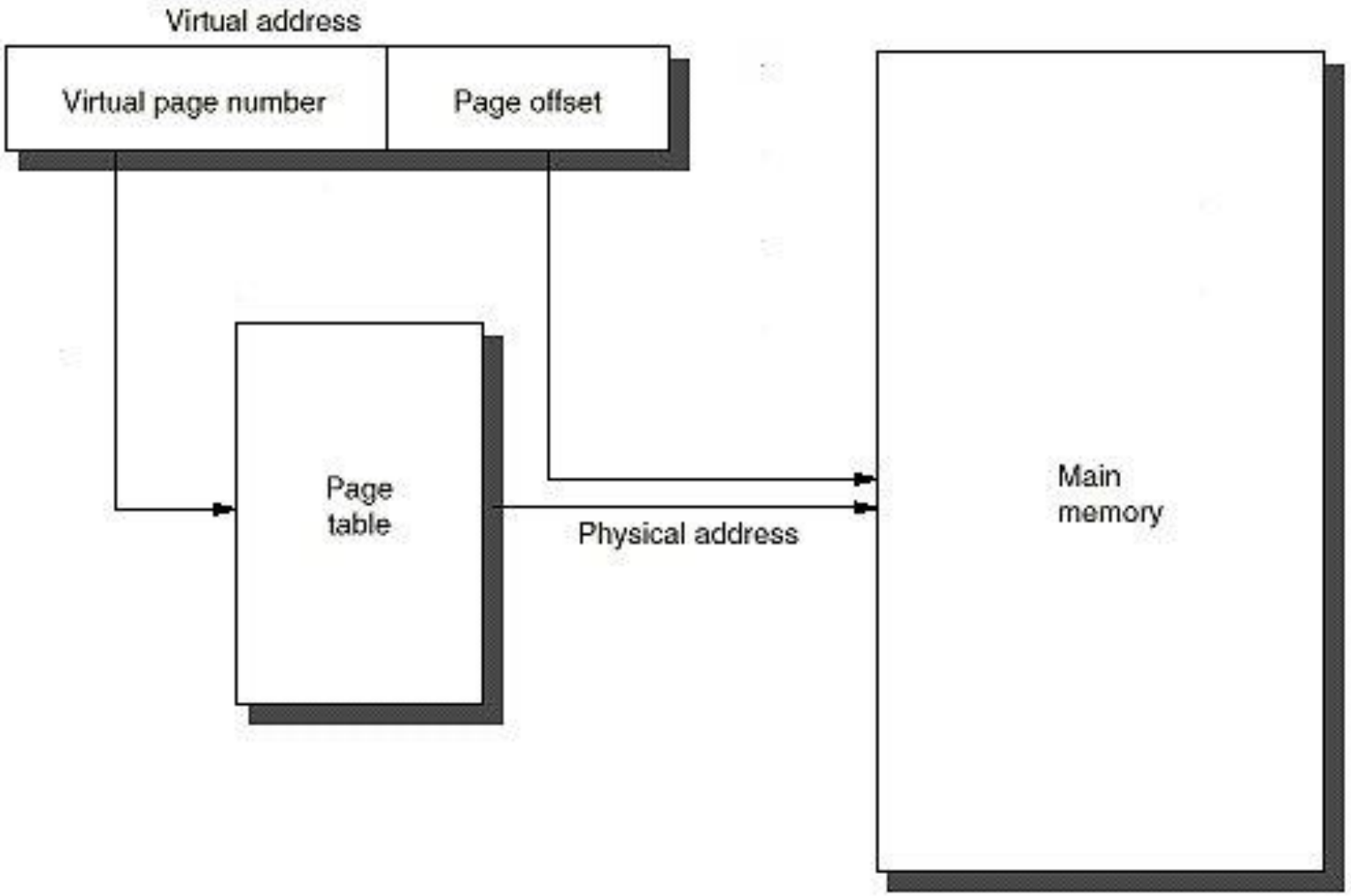
Page Table

Χρειάζονται 2 προσπελάσεις μνήμης:

- **στο page table**
- **στο αντικείμενο**



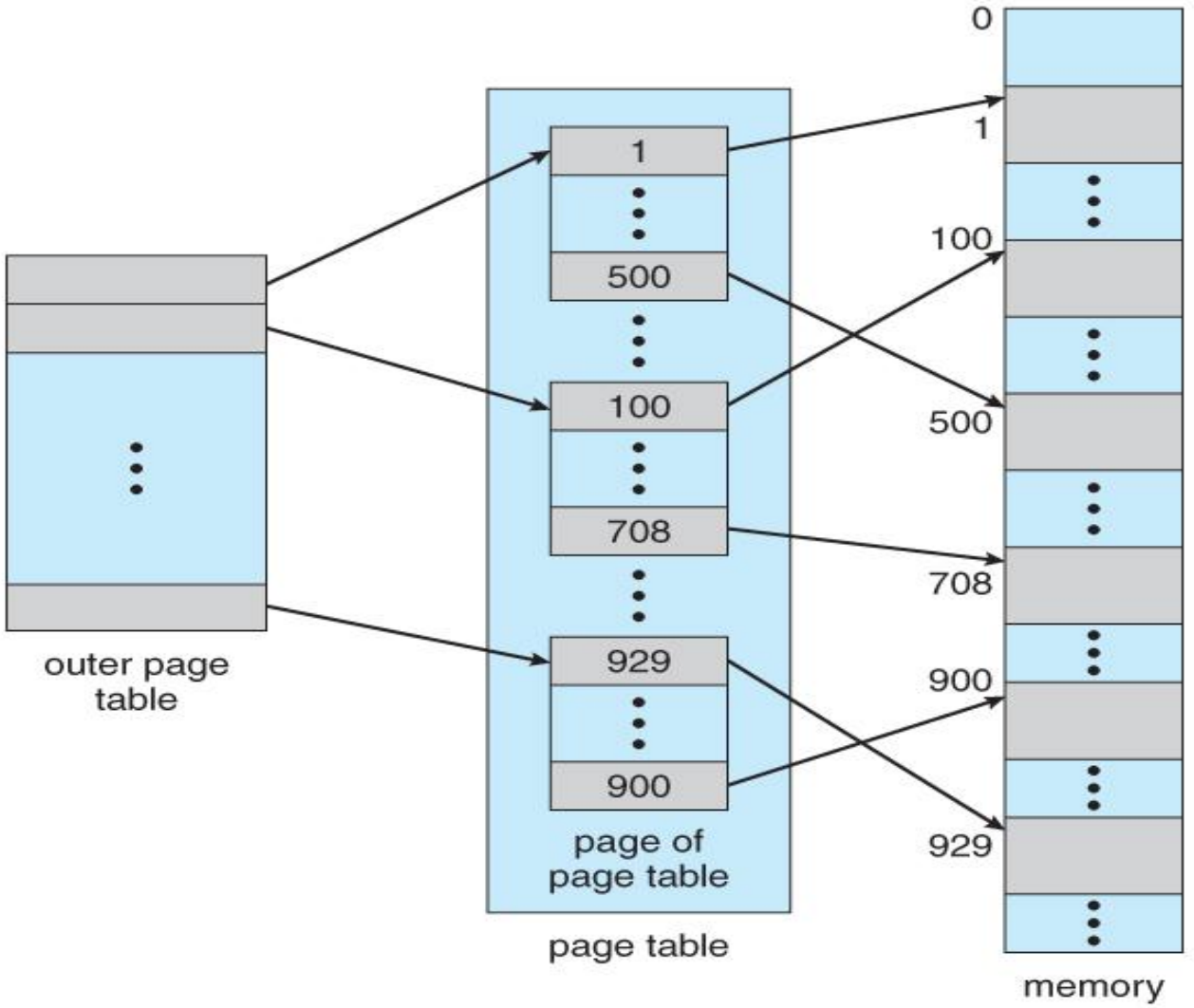
Αντιστοίχιση των Virtual Addresses σε Physical Addresses μέσω ενός πίνακα σελίδων (Page Table)



Hierarchical Page Tables

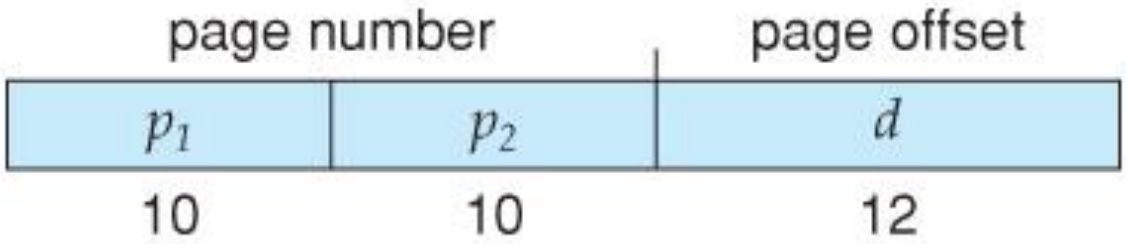
- Multilevel Paging
- Κατάτμηση του χώρου διευθύνσεων σε πολλαπλά page tables
- Two-level, three-level, etc
- Τα τμήματα του πίνακα σελίδων που δεν χρειάζονται δεν βρίσκονται στη μνήμη.

Two-Level Page Table (1)



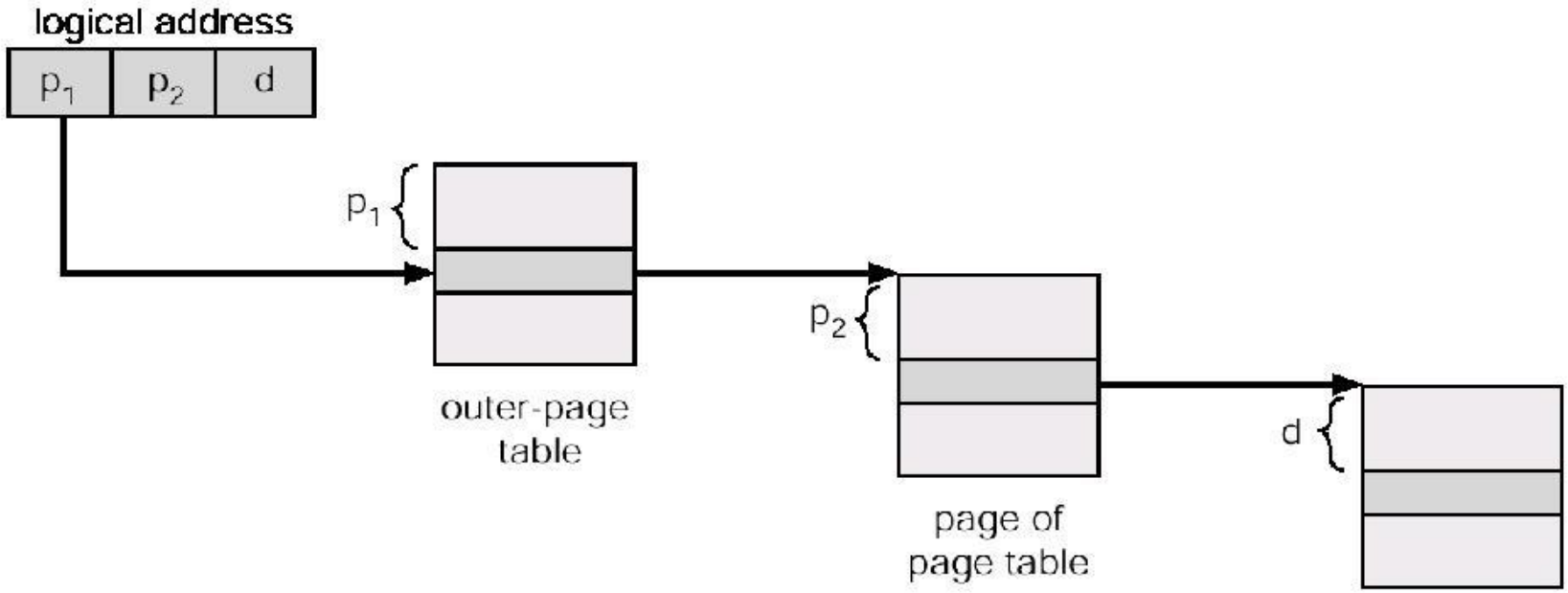
Two-Level Page Table (2)

- 32-bit machine, 4K page size:
  - page number : 20 bits
  - page offset : 12 bits
- Σελιδοποιούμε και τον page table:
  - p_1 10-bit page number
  - p_2 10-bit page offset



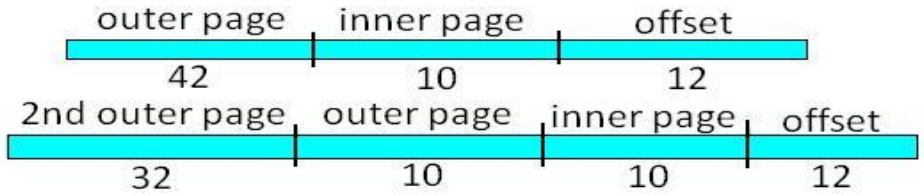
Two-Level Page Table (3)

- Μετάφραση διεύθυνσης

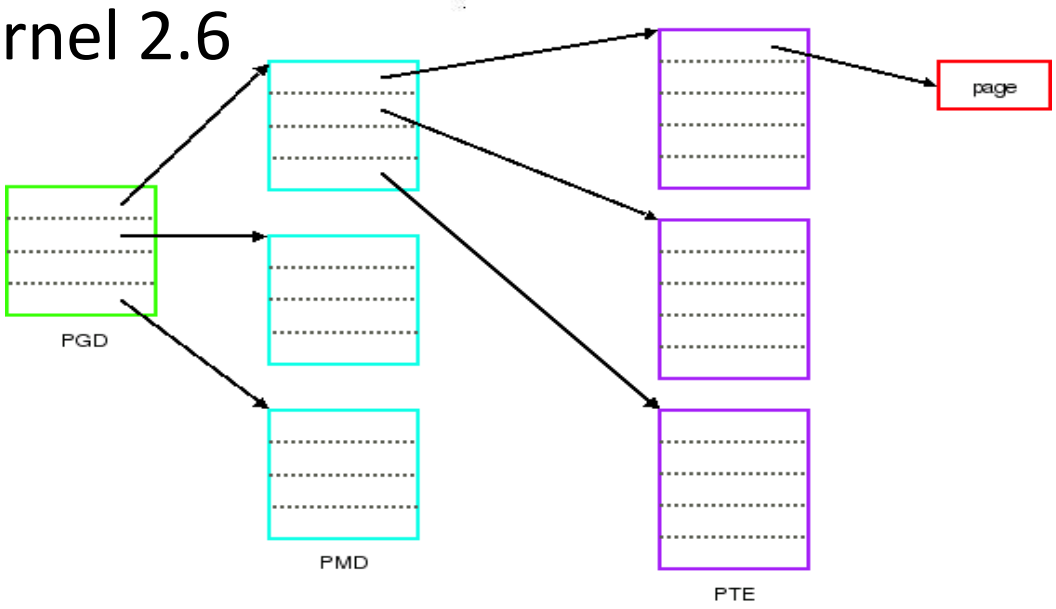


Three-Level Page Table

- 64-bit machine, 4K page size



- x86 system, PAE mode, kernel 2.6
  - PGD 4 entries → 2 bits (30-31)
  - PMD 512 entries → 9 bits (21-29)
  - PTE 512 entries → 9 bits (12-20)
  - page offset → 12 bits



- four-level, x86_64 (48-bit virtual address)
  - PML4 512 entries → 9 bits
  - PGD 512 entries → 9 bits
  - PMD 512 entries → 9 bits
  - PTE 512 entries → 9 bits
  - page offset → 12 bits

Address Length> 48 bits ⟶ PML5 ?!?

Multi-level Page Table & Translation

- Πλεονεκτήματα
  - Δέσμευση μόνο όσων page table entries είναι απαραίτητες για την εφαρμογή
  - Εύκολη δέσμευση μνήμης (memory allocation)
  - Εύκολος διαμοιρασμός (σε επίπεδο segment ή page).
- Μειονεκτήματα
  - Ένας δείκτης για κάθε σελίδα
  - Contiguous page tables
  - 2 lookups (για 2-level, n για n-level) για κάθε reference
    - Ακριβό!

Inverted Page Table - Κίνητρο

- Παράδειγμα:
  - 64-bit machine
  - 4K page size
  - 512M φυσική μνήμη
- Πόση μνήμη χρειαζόμαστε για την αποθήκευση του πίνακα σελίδων?

Single-level Page Table

- Μία εγγραφή στον πίνακα σελίδων ανά εικονική σελίδα
  - Εικονική διεύθυνση 64 bits, 12 bits offset = **2^{52} καταχωρήσεις στον πίνακα σελίδων**
- Μέγεθος εγγραφής πίνακα σελίδων
  - Κάθε εγγραφή περιέχει:
    - Access control bits + Physical page number
  - 512M φυσική μνήμη = 2^{29} bytes
    - Φυσική διεύθυνση 29 bits, 12 bits offset = 2^{17} σελίδες
    - Χρειάζονται **17 bits** για το physical page number
  - Μέγεθος εγγραφής πίνακα σελίδων ≈ **4 bytes**
    - 17 bits physical page number ≈ 3 bytes
    - Access control bits ≈ 1 byte
- Μέγεθος πίνακα σελίδων
  - 2^{52} εγγραφές × 2^2 bytes = **2^{54} bytes (16 Petabytes)**

- 64-bit machine
- 4K page size
- 512M φυσική μνήμη

Πολλή μνήμη!
Ανά διεργασία...

Multi-level Page Table

- Πόσα επίπεδα χρειαζόμαστε για να εξασφαλίσουμε ότι κάθε πίνακας σελίδων απαιτεί μόνο μία σελίδα (4K)?
  - Κάθε εγγραφή του page table απαιτεί 4 bytes
  - Κάθε επίπεδο θα έχει 4K/4 bytes = 1024 εγγραφές
  - $1024 = 2^{10} = 10$ bits ανά επίπεδο
  - Για 52 bits χρειαζόμαστε 6 επίπεδα

- 64-bit machine
- 4K page size
- 512M φυσική μνήμη

6 επίπεδα → 6 προσβάσεις στη μνήμη → αργό!

Παρατήρηση

- 512M φυσικής μνήμης
  - 2^{29} bytes/2^{12} offset = 2^{17} φυσικές σελίδες
- Υπάρχει τρόπος να αποθηκεύσουμε μόνο μία εγγραφή ανά φυσική σελίδα στον πίνακα σελίδων?
  - Μπορούμε να μειώσουμε το μέγεθος του πίνακα σελίδων ακόμα και στα 2M.
    - Θεωρώντας ότι κάθε εγγραφή έχει μήκος 16 bytes
      - 2^{17} εγγραφές × 2^4 bytes = 2^{21} = **2M**

Inverted Page Table

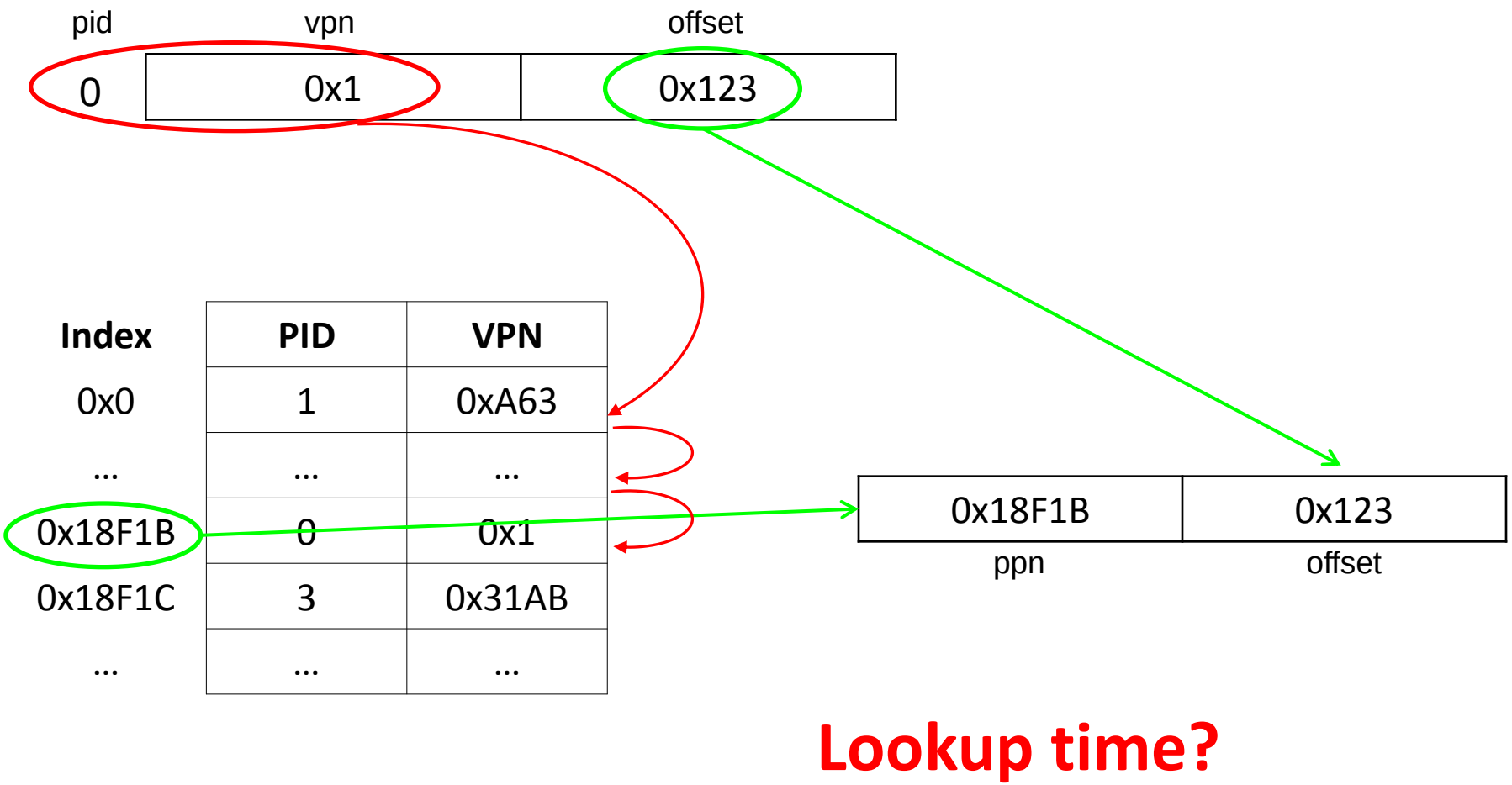
Χρειάζεται μόνο ένας καθολικός πίνακας σελίδων (όχι ανά διεργασία)

Inverted Page Table (1)

- Συνδυασμός page table και frame table σε μια δομή.
- Κάθε εγγραφή του πίνακα περιέχει
  - Process ID (πίνακας σελίδων είναι κοινός για όλες τις διεργασίες)
  - Virtual page number
- Μία εγγραφή για κάθε φυσική σελίδα
- Στο παράδειγμά μας:
  - Έστω 16 bits για το process ID, 52 bits virtual page number, 12 access bits
    - 80 bits σύνολο = 10 bytes
    - 10 bytes × 2^{17} εγγραφές ≈ **1.3M** (για όλο το σύστημα!)

- 64-bit machine
- 4K page size
- 512M φυσική μνήμη

Inverted Page Table (2)

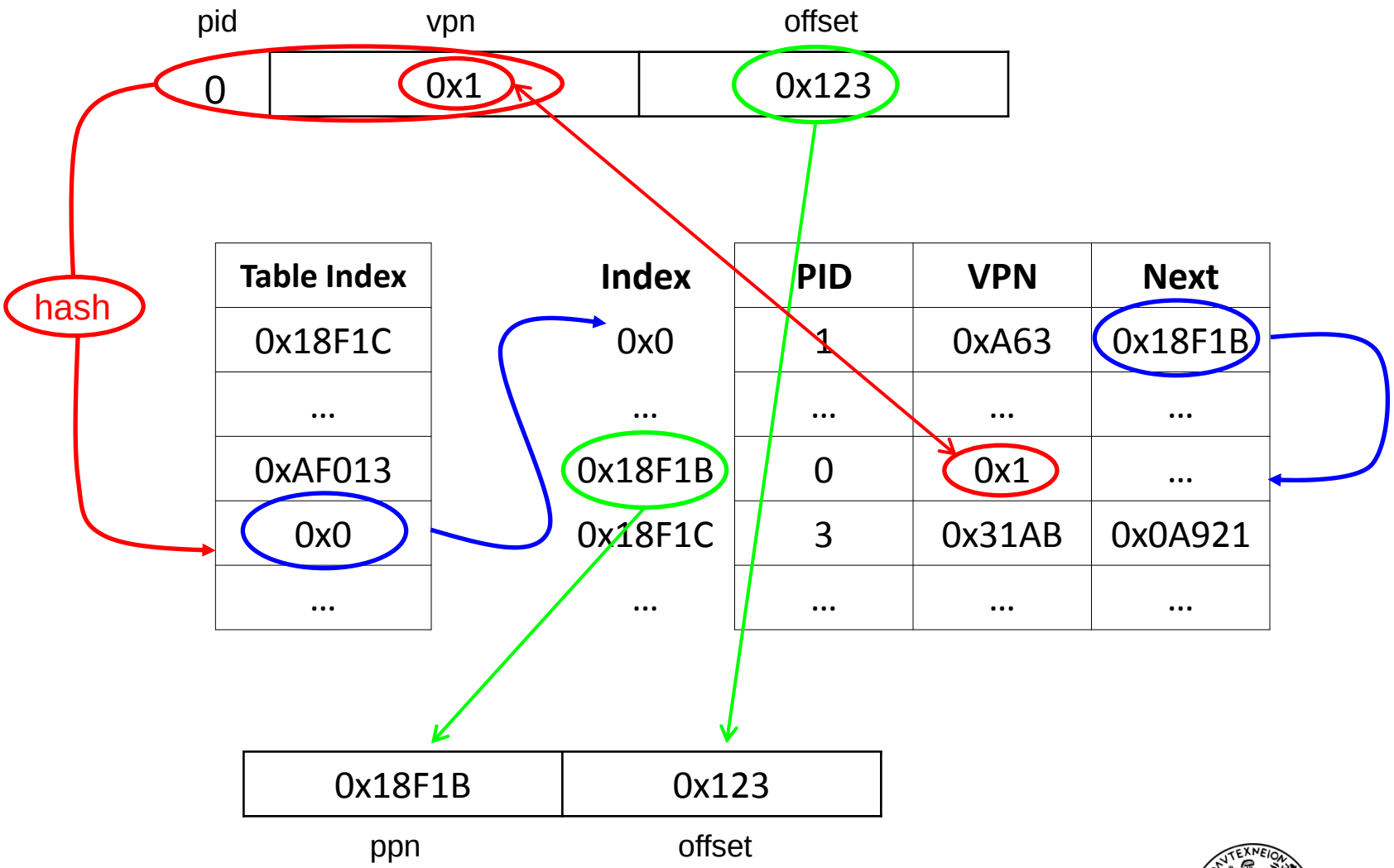


Lookup time?

Hashed Inverted Page Table (1)

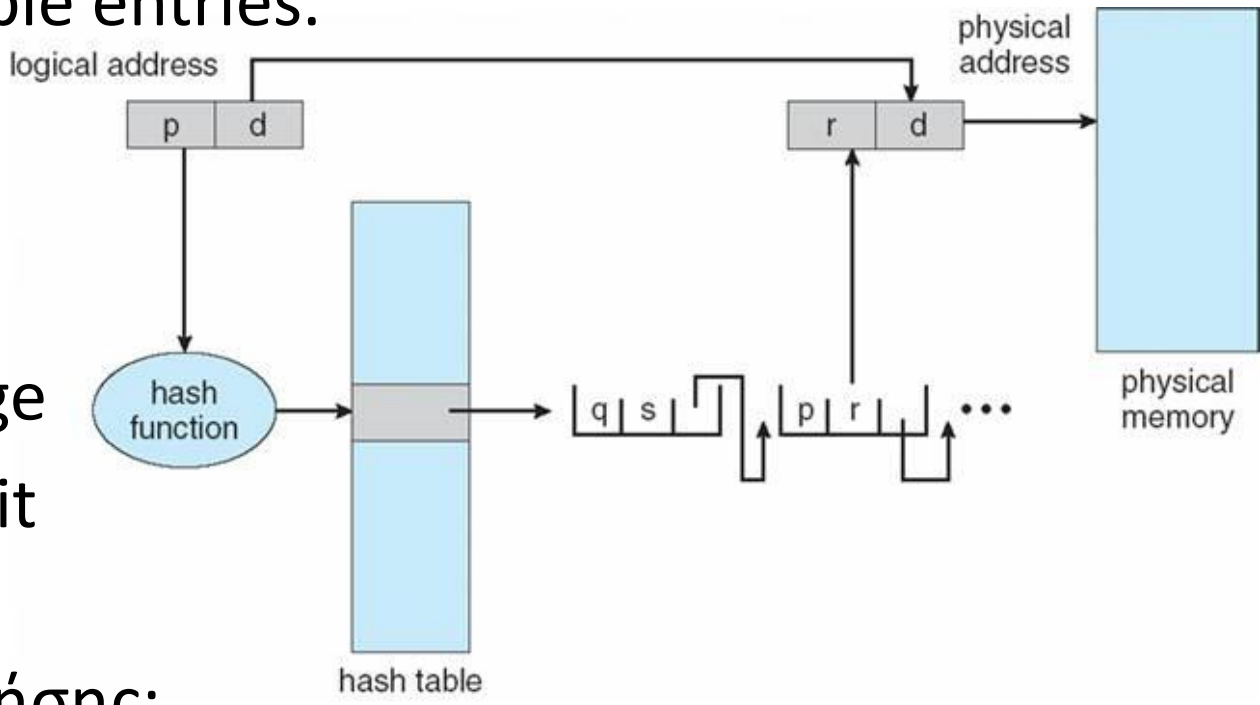
- Το Inverted Page Table απαιτεί πολλές προσβάσεις μνήμης για το lookup.
- Μπορούμε να κρατάμε ένα επιπλέον επίπεδο πριν το πραγματικό page table (hash anchor table)
  - Αντιστοιχίζει ζεύγη process ID, virtual page number σε εγγραφές του πίνακα σελίδων
    - Χρήση πεδίου *Next* στον πίνακα σελίδων για επίλυση συγκρούσεων
- Πηγαίνουμε πρώτα στον hash anchor table
  - Σύγκριση process ID και virtual page number
    - Αν ταιριάζουν τότε βρέθηκε η κατάλληλη εγγραφή
    - Αν όχι ακολουθούμε το πεδίο *Next* μέχρι να φτάσουμε στην κατάλληλη εγγραφή

Hashed Inverted Page Table (2)

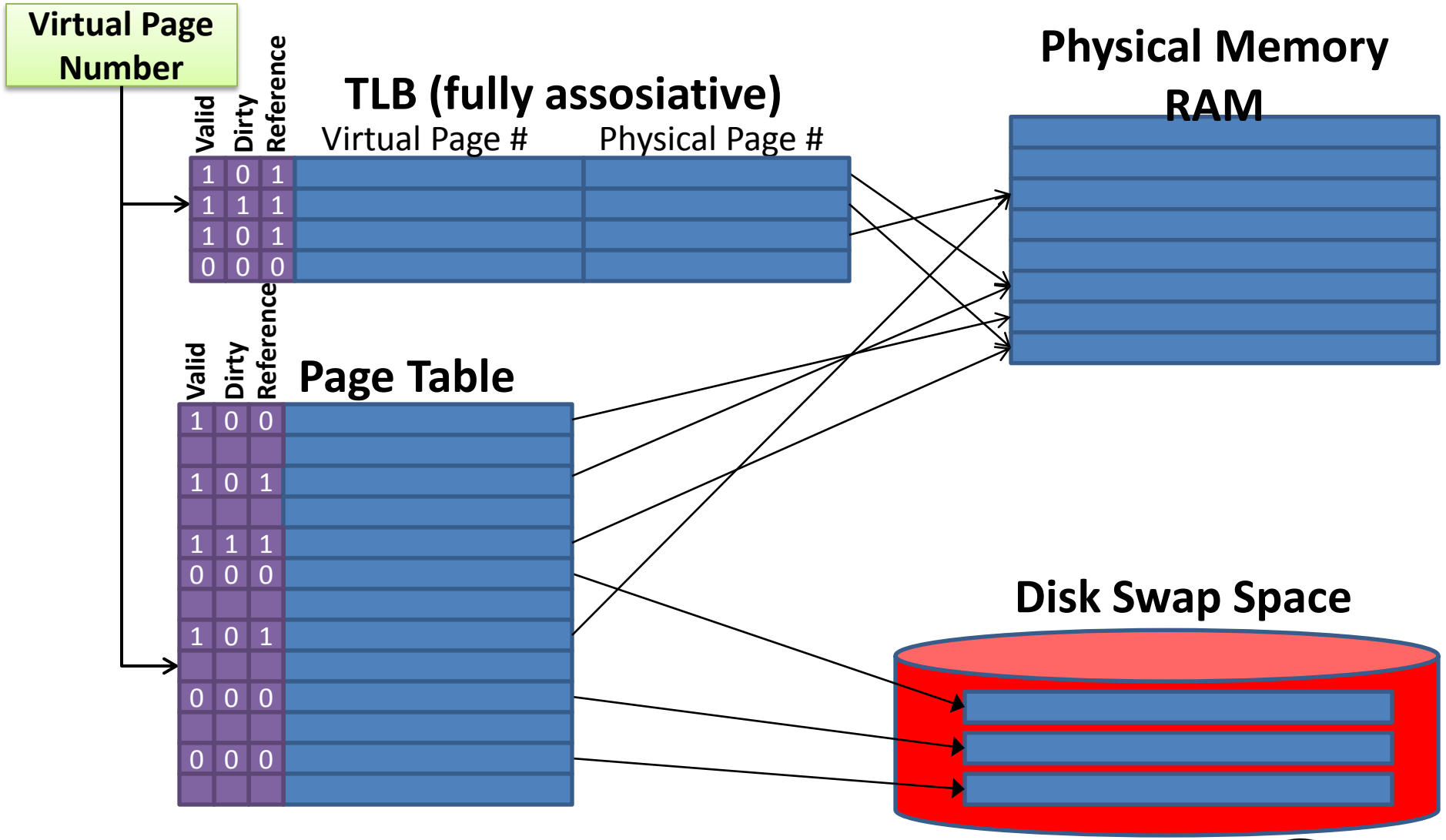


Hashed Inverted Page Table (3)

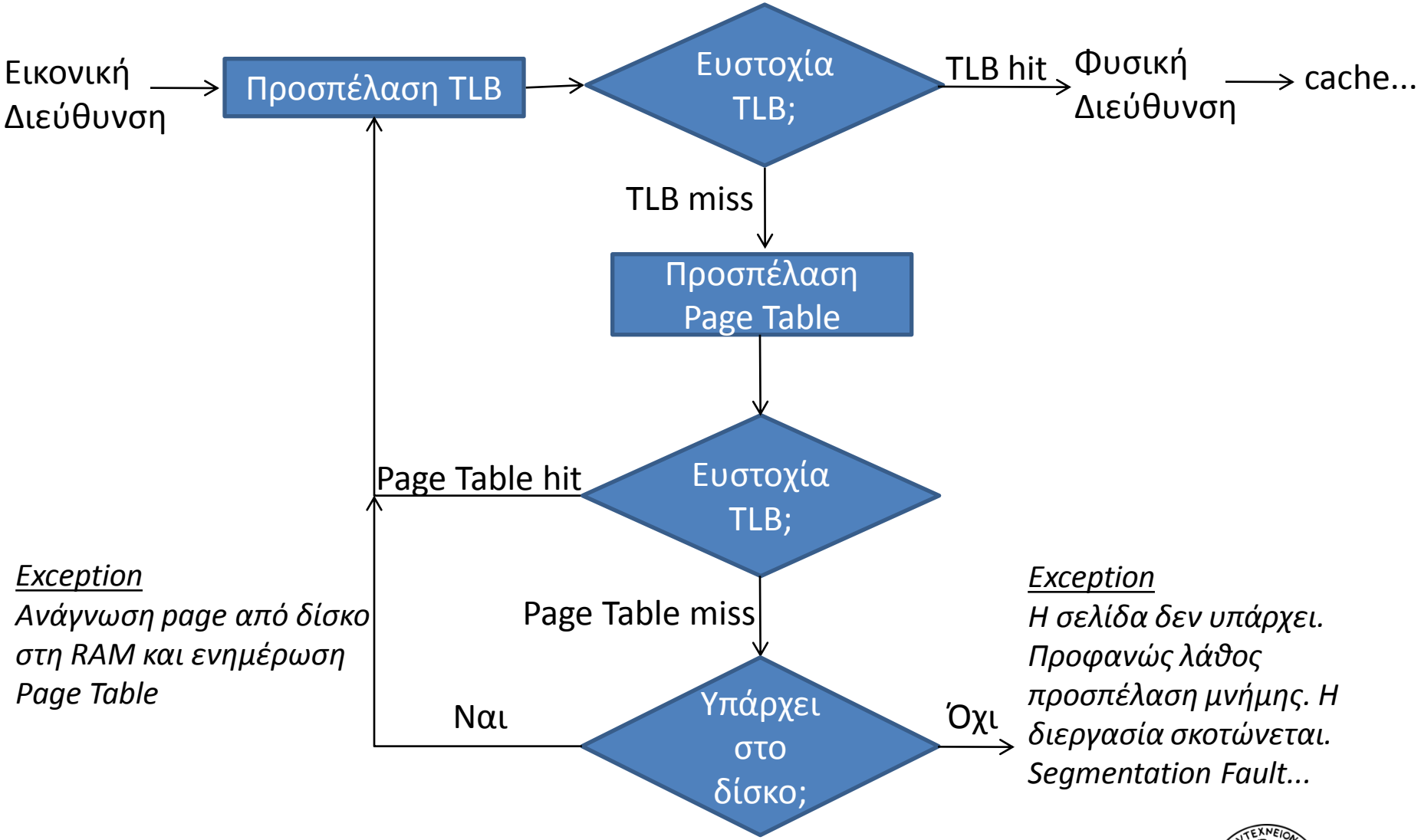
- Χρήση hash tables για τον περιορισμό της αναζήτησης σε 1 (ή λίγα) page table entries.
- Collision chains
- Hash function: Optimised for speed not coverage
- Θελκτικό για 64-bit αρχιτεκτονικές
- Παραδείγματα χρήσης:
  - PowerPC, UltraSPARC, IA-64

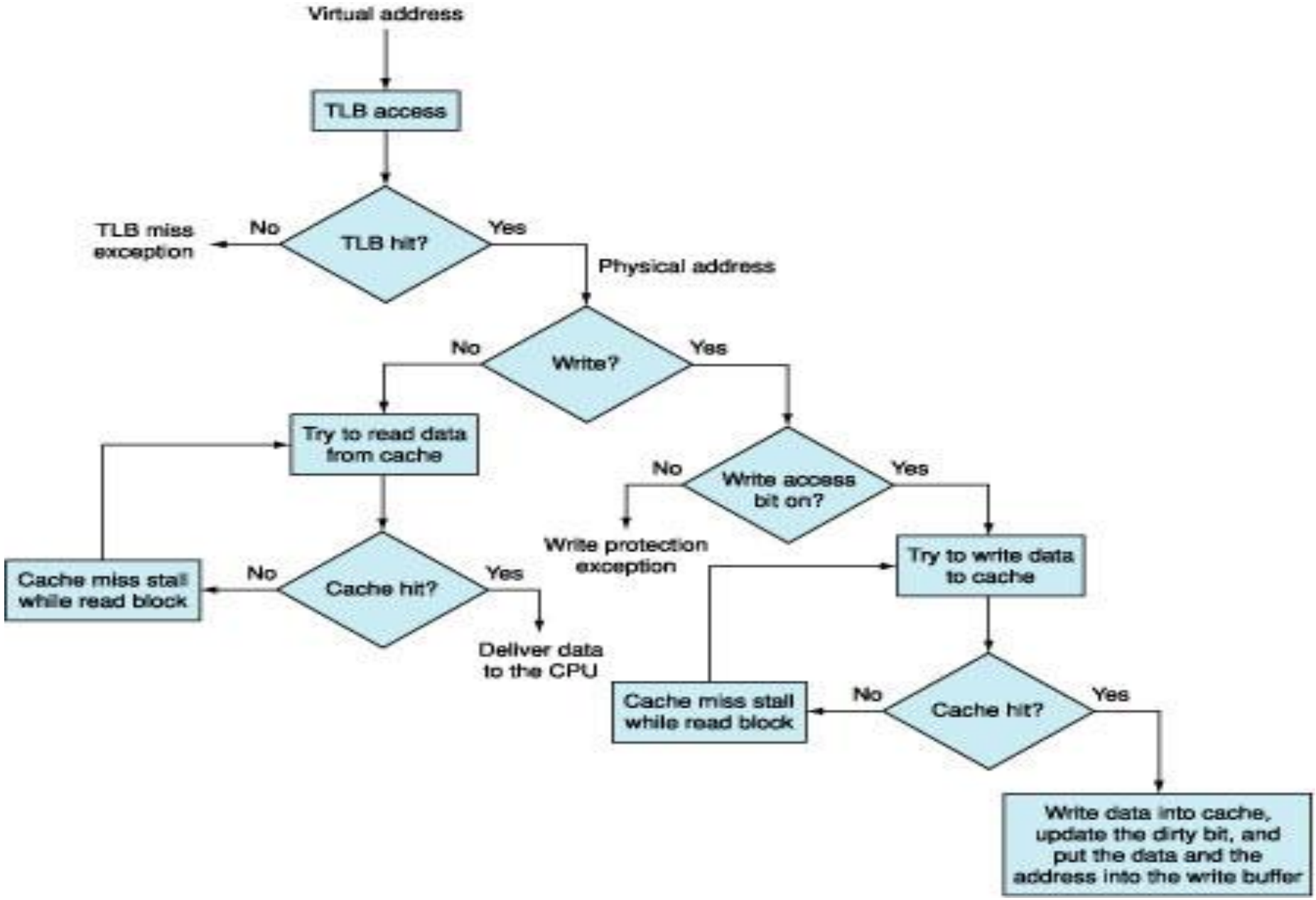


Translation-Lookaside Buffer (TLB)

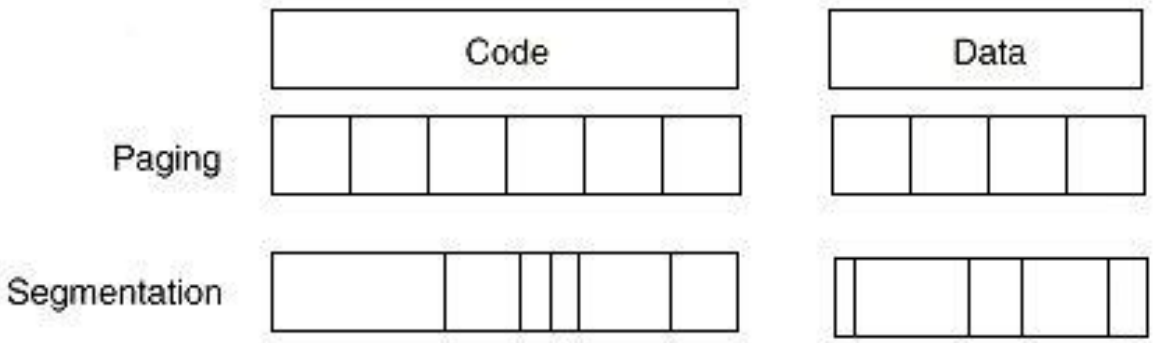


Ενέργειες Μετάφρασης Διευθύνσεων




Virtual address
TLB access
TLB hit?
No
TLB miss exception
Yes
Physical address
Write?
No
Yes
Try to read data from cache
Write access bit on?
No
Yes
Write protection exception
Cache miss stall while read block
Cache hit?
No
Yes
Deliver data to the CPU
Try to write data to cache
Cache miss stall while read block
Cache hit?
No
Yes
Write data into cache, update the dirty bit, and put the data and the address into the write buffer

Paging & Segmentation



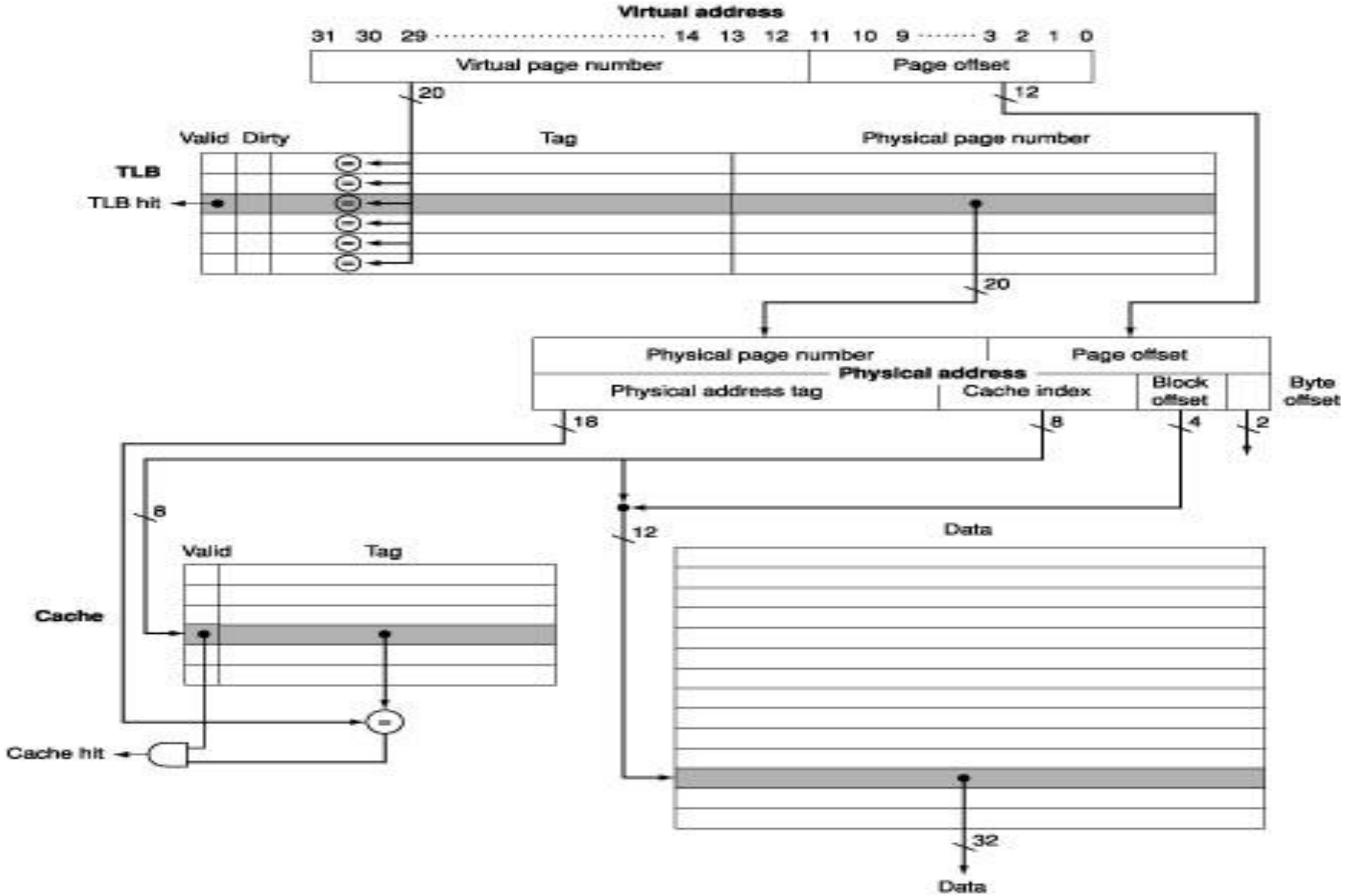
	Page	Segment
Words per address	One	Two (segment and offset)
Programmer visible?	Invisible to application programmer	May be visible to application programmer
Replacing a block	Trivial (all blocks are the same size)	Hard (must find contiguous, variable-size, unused portion of main memory)
Memory use inefficiency	Internal fragmentation (unused portion of page)	External fragmentation (unused pieces of main memory)
Efficient disk traffic	Yes (adjust page size to balance access time and transfer time)	Not always (small segments may transfer just a few bytes)

Τυπικές παράμετροι Cache και Virtual Memory

Parameter	First-level cache	Virtual memory
Block (page) size	16–128 bytes	4096–65,536 bytes
Hit time	1–2 clock cycles	40–100 clock cycles
Miss penalty	8–100 clock cycles	700,000–6,000,000 clock cycles
(Access time)	(6–60 clock cycles)	(500,000–4,000,000 clock cycles)
(Transfer time)	(2–40 clock cycles)	(200,000–2,000,000 clock cycles)
Miss rate	0.5–10%	0.00001– 0.001%
Data memory size	0.016–1MB	16–8192 MB

Virtual Memory: Στρατηγικές

- Τοποθέτηση του block στην κύρια μνήμη: Η fully associative τεχνική χρησιμοποιείται για την ελάττωση του miss rate.
- Αντικατάσταση του block: The least recently used (LRU) block αντικαθίσταται όταν ένα νέο block έρχεται στη μνήμη από το δίσκο.
- Στρατηγική εγγραφών: Χρησιμοποιείται η τεχνική write back και μόνο οι dirty σελίδες μεταφέρονται από την κύρια μνήμη στο δίσκο.
- Για την τοποθέτηση των blocks στην κύρια μνήμη χρησιμοποιείται ένας page table. Ο page table δεικτοδοτείται από τον εικονικό αριθμό σελίδας (virtual page number) και περιέχει τη φυσική διεύθυνση (physical address) του block.
  - **Paging: Το Offset συγχωνεύεται με τη διεύθυνση της φυσικής σελίδας.**
  - **Segmentation: Το Offset προστίθεται στη διεύθυνση του physical segment.**
- Για την αξιοποίηση της address locality, χρησιμοποιείται συνήθως ο translation look-aside buffer (TLB) για την αποθήκευση των προσφάτως μεταφρασμένων διευθύνσεων ώστε να αποφεύγεται προσπέλαση της μνήμης προκειμένου να διαβαστεί ο πίνακας σελίδων (page table).


Virtual address
31 30 29 ..... 14 13 12 11 10 9 ..... 3 2 1 0
Virtual page number
Page offset
20
12
Valid Dirty
Tag
Physical page number
TLB
TLB hit
20
Physical page number
Page offset
Physical address
Physical address tag
Cache index
Block offset
Byte offset
18
8
4
2
6
12
Data
Valid
Tag
Cache
Cache hit
32
Data

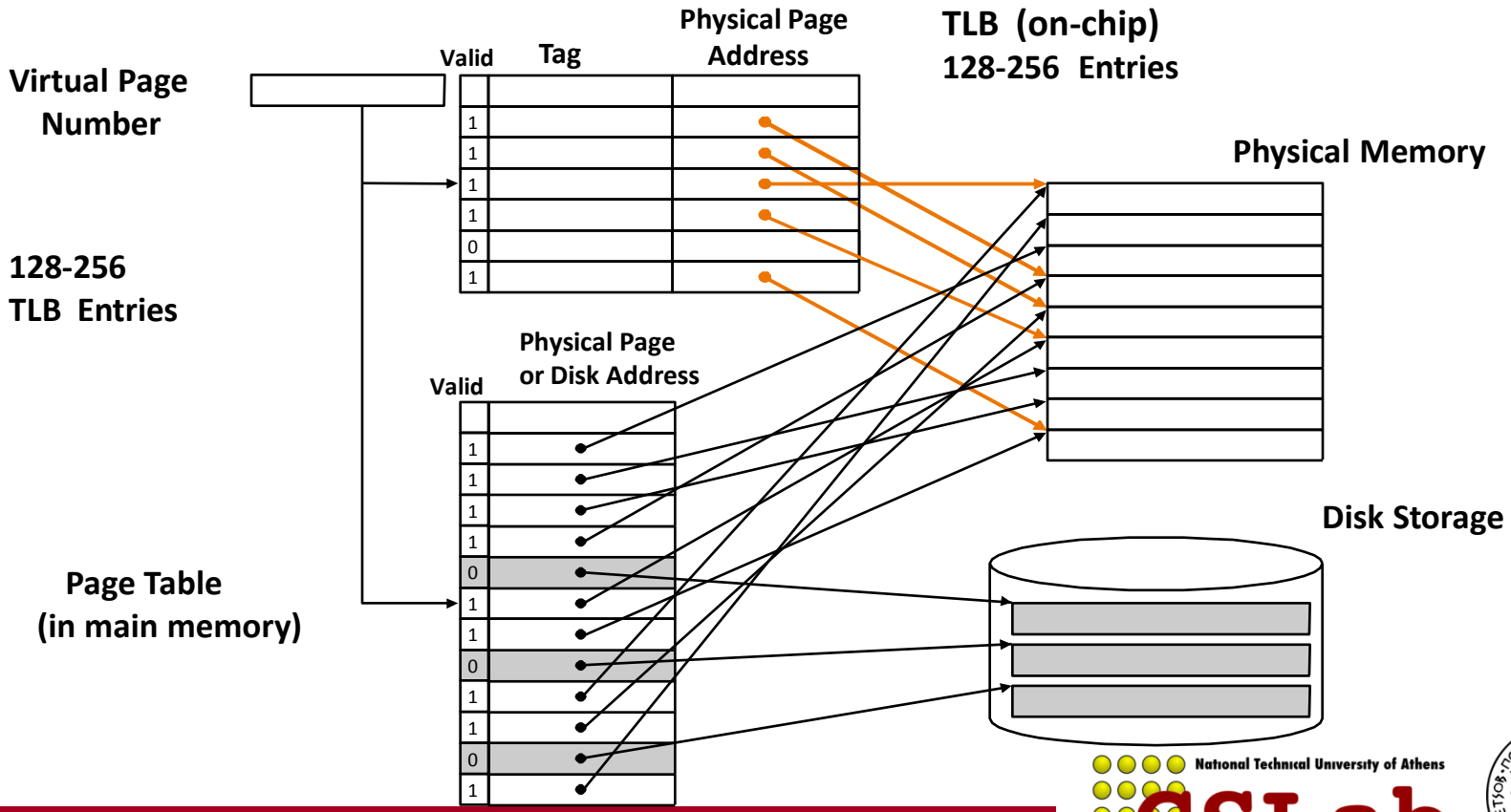
Page Faults

- Page faults: the data is not in memory, retrieve it from disk
  - huge miss penalty, thus pages should be fairly large (e.g., 4KB)
  - reducing page faults is important (LRU is worth the price)
  - can handle the faults in software instead of hardware
  - using write-through is too expensive so we use writeback

Επιτάχυνση της μετάφρασης διευθύνσεων

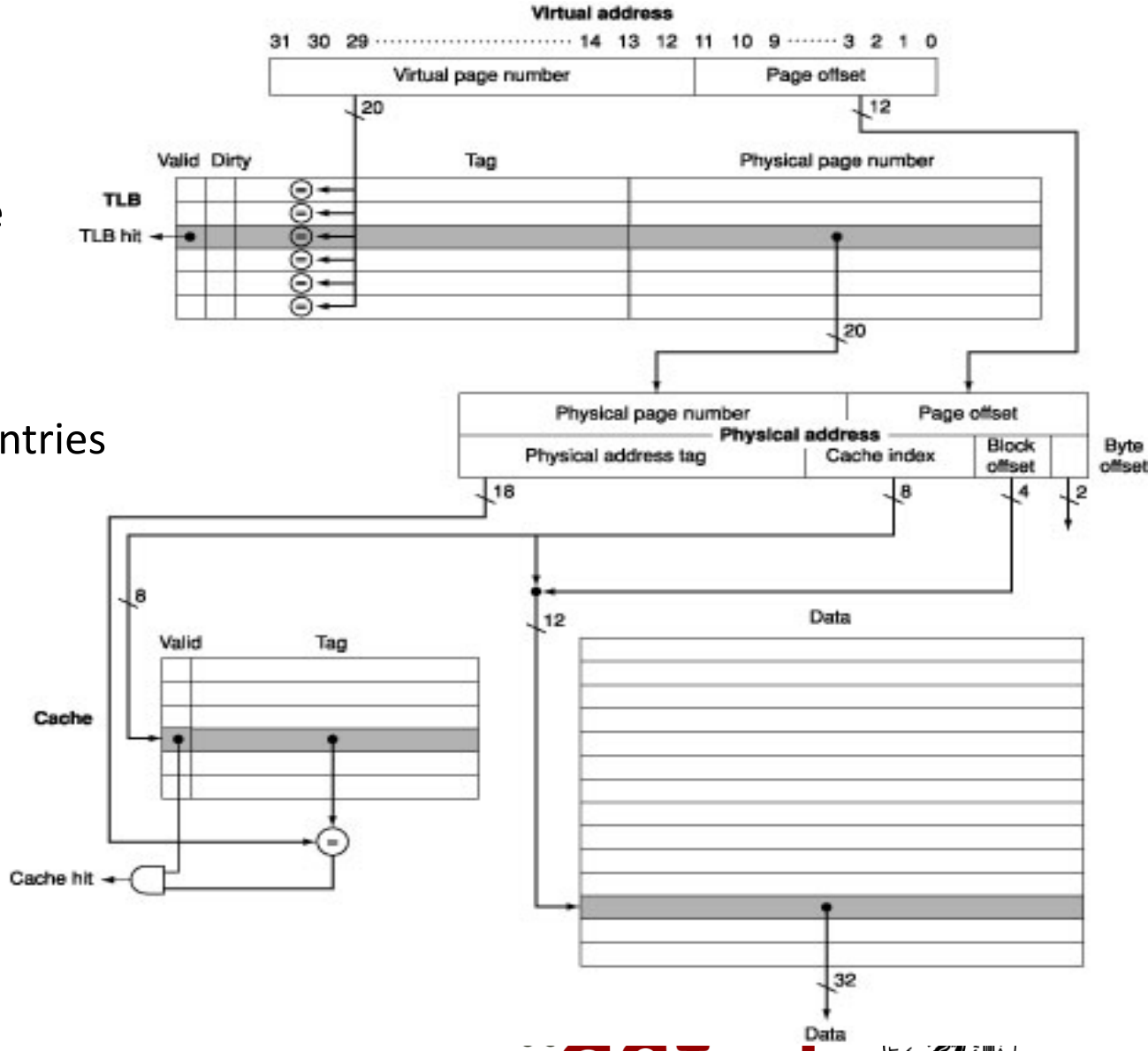
Translation Lookaside Buffer (TLB)

- TLB: Μία μικρή on-chip fully-associative cache που χρησιμοποιείται για τη μετάφραση διευθύνσεων.
- Αν μία virtual address υπάρχει μέσα στο TLB (TLB hit), δεν προσπελάζεται ο πίνακας σελίδων της κύριας μνήμης.



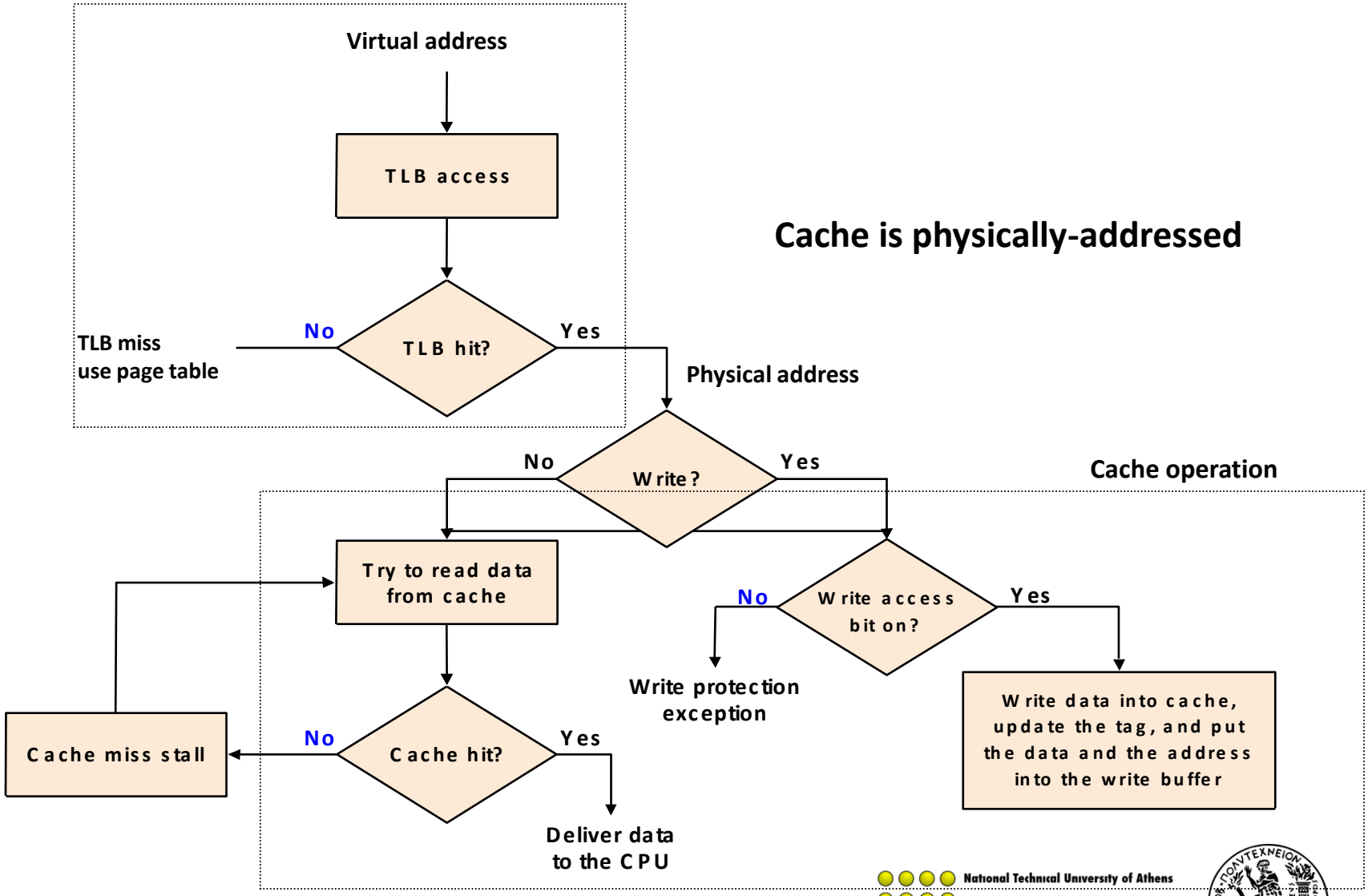
Intrinsity fastMATH processor

- 32 bit address space/byte addressing
- 4KB/page
- TLB fully associative: 16 entries



TLB & Cache Operation (Intrinsity FastMATH)

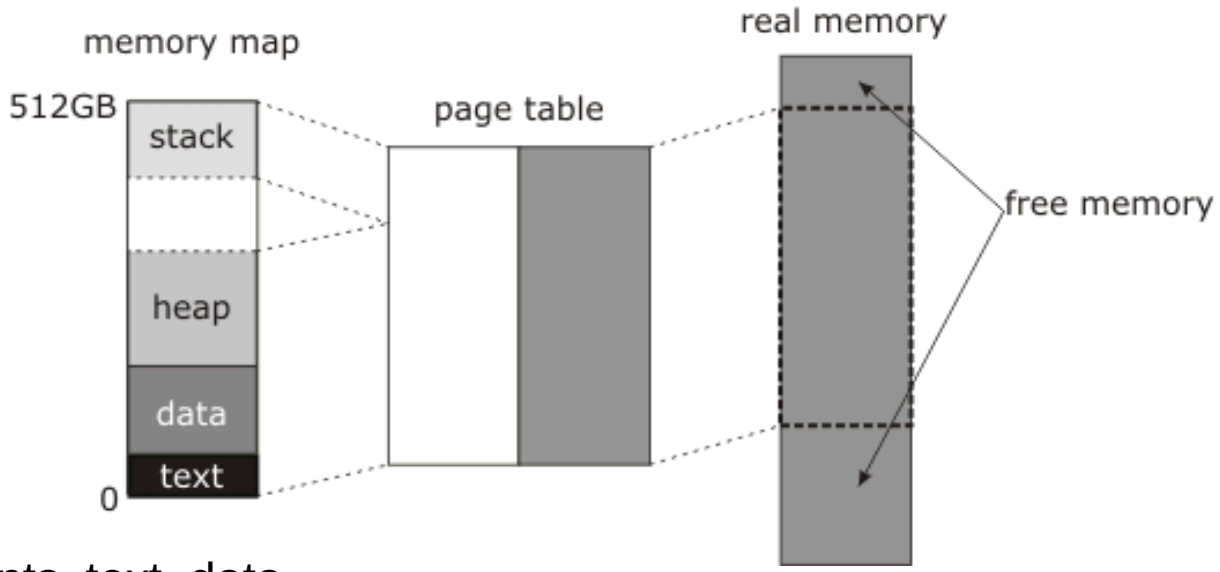
TLB Operation



TLB, virtual memory, cache συνδυασμοί

TLB	Page Table	Cache	Possible? If so under what circumstance?
hit	hit	miss	Possible, although the page table is never really checked if TLB hits
miss	hit	hit	TLB misses, but entry found in page table; after retry, data is found in cache
miss	hit	miss	TLB misses, but entry found in page table; after retry data misses in cache
miss	miss	miss	TLB misses and is followed by a page fault; after retry, data must miss in cache
hit	miss	miss	Impossible: cannot have a translation in TLB if page is not present in memory
hit	miss	hit	Impossible: cannot have a translation in TLB if page is not present in memory
miss	miss	hit	Impossible: data cannot be allowed in cache if the page is not in memory

Memory map, page table, physical memory



All three segments, text, data (data+bss), and heap, are mapped to real memory through the page table.

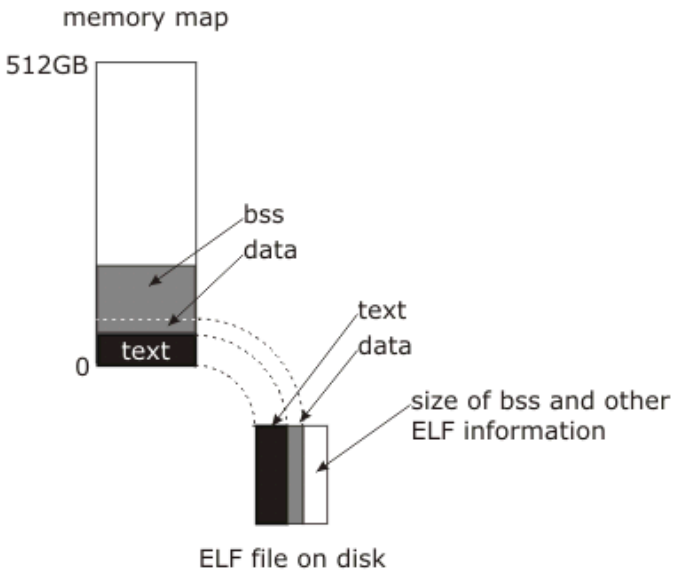
The figure shows that the heap segment expands and contracts as memory is allocated and deallocated.

Consequently, page table entries are added or deleted as necessary.

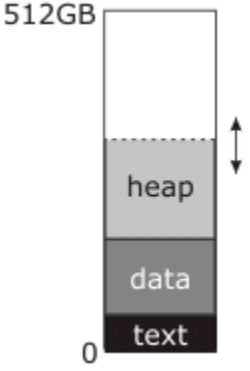
$ cat /proc/23114/maps

(a) 00400000-00401000 r-xp ... /scratch/esumbar/map
(b) 00500000-00501000 rw-p ... /scratch/esumbar/map
(c) 00501000-02d01000 rwxp ...
(d) 2a95556000-2a95557000 rw-p ...
(e) 2a95578000-2a9697c000 rw-p ...
(f) 330c300000-330c315000 r-xp ... /lib64/ld-2.3.4.so
(g) 330c414000-330c416000 rw-p ... /lib64/ld-2.3.4.so
(h) 330c500000-330c62a000 r-xp ... /lib64/tls/libc-2.3.4.so
(i) 330c62a000-330c729000 ---p ...

Process memory map

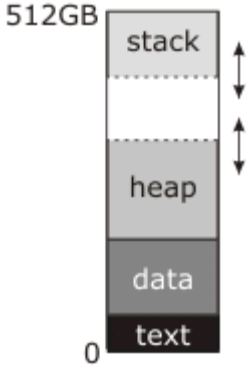


showing text, data, and bss segments.



showing heap

heap segment expands and contracts as memory is allocated and deallocated. page table entries are added or deleted as necessary.



showing stack for subroutine calls

Pentium P4 / AMD Opteron

Characteristic	Intel Pentium P4	AMD Opteron
Virtual address	32 bits	48 bits
Physical address	36 bits	40 bits
Page size	4 KB, 2/4 MB	4 KB, 2/4 MB
TLB organization	1 TLB for instructions and 1 TLB for data Both are four-way set associative Both use pseudo-LRU replacement Both have 128 entries TLB misses handled in hardware	2 TLBs for instructions and 2 TLBs for data Both L1 TLBs fully associative, LRU replacement Both L2 TLBs are four-way set associativity, round-robin LRU Both L1 TLBs have 40 entries Both L2 TLBs have 512 entries TLB misses handled in hardware

FIGURE 7.34 Address translation and TLB hardware for the Intel Pentium P4 and AMD Opteron. The word size sets the maximum size of the virtual address, but a processor need not use all bits. The physical address size is independent of word size. The P4 has one TLB for instructions and a separate identical TLB for data, while the Opteron has both an L1 TLB and an L2 TLB for instructions and identical L1 and L2 TLBs for data. Both processors provide support for large pages, which are used for things like the operating system or mapping a frame buffer. The large-page scheme avoids using a large number of entries to map a single object that is always present.

Pentium P4 / AMD Opteron

Characteristic	Intel Pentium P4	AMD Opteron
L1 cache organization	Split instruction and data caches	Split instruction and data caches
L1 cache size	8 KB for data, 96 KB trace cache for RISC instructions (12K RISC operations)	64 KB each for instructions/data
L1 cache associativity	4-way set associative	2-way set associative
L1 replacement	Approximated LRU replacement	LRU replacement
L1 block size	64 bytes	64 bytes
L1 write policy	Write-through	Write-back
L2 cache organization	Unified (instruction and data)	Unified (instruction and data)
L2 cache size	512 KB	1024 KB (1 MB)
L2 cache associativity	8-way set associative	16-way set associative
L2 replacement	Approximated LRU replacement	Approximated LRU replacement
L2 block size	128 bytes	64 bytes
L2 write policy	Write-back	Write-back

FIGURE 7.35 First-level and second-level caches in the Intel Pentium P4 and AMD Opteron. The primary caches in the P4 are physically indexed and tagged; for a discussion of the alternatives, see the Elaboration on page 527.

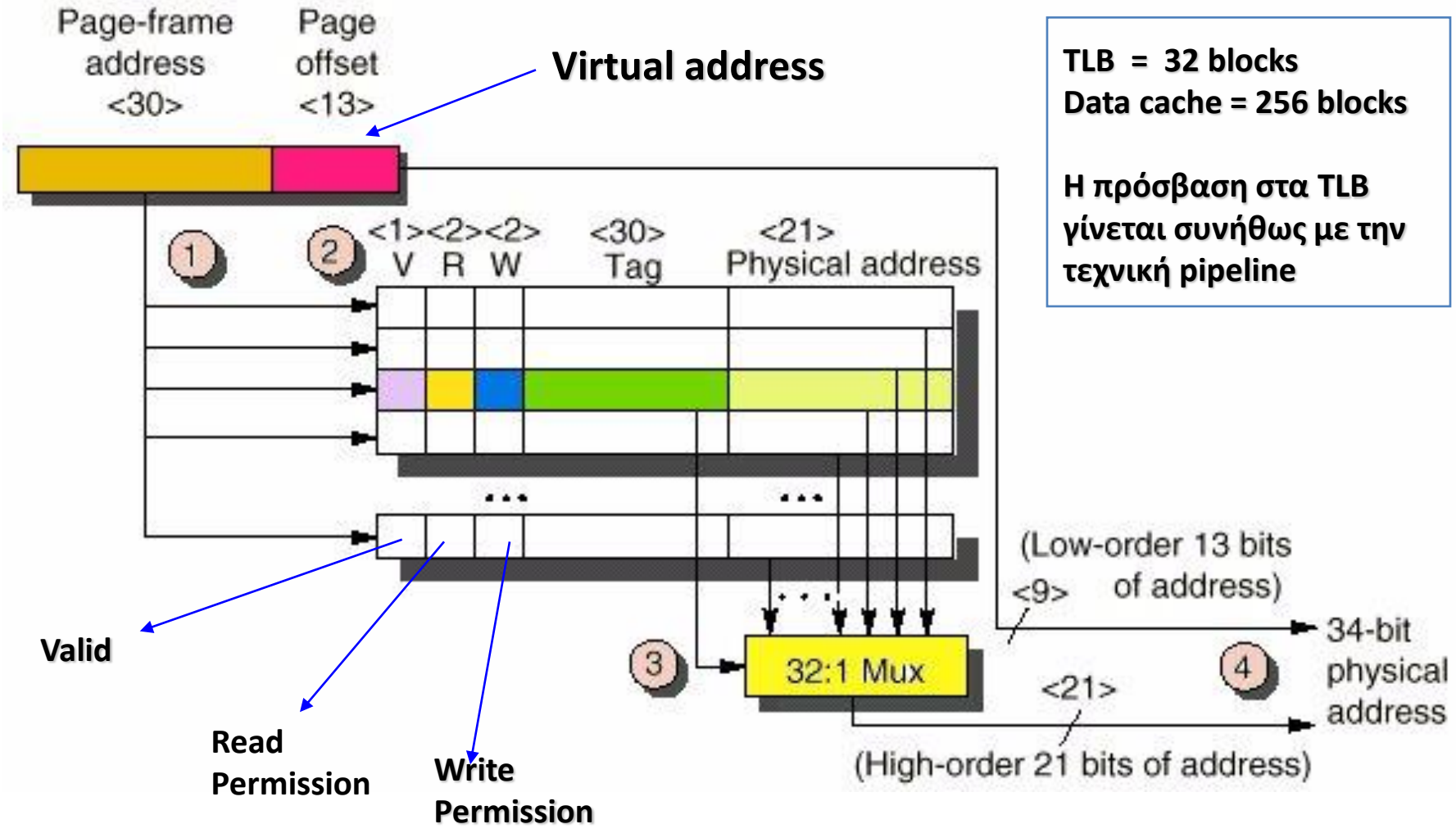
Desktop/embedded/server microprocessors (2004)

- Things are getting complicated!

MPU	AMD Opteron	Intrinsity FastMATH	Intel Pentium 4	Intel PXA250	Sun UltraSPARC IV
Instruction set architecture	IA-32, AMD64	MIPS32	IA-32	ARM	SPARC v9
Intended application	server	embedded	desktop	low-power embedded	server
Die size (mm^2) (2004)	193	122	217		356
Instructions issued/clock	3	2	3 RISC ops	1	4 × 2
Clock rate (2004)	2.0 GHz	2.0 GHz	3.2 GHz	0.4 GHz	1.2 GHz
Instruction cache	64 KB, 2-way set associative	16 KB, direct mapped	12000 RISC op trace cache (~96 KB)	32 KB, 32-way set associative	32 KB, 4-way set associative
Latency (clocks)	3?	4	4	1	2
Data cache	64 KB, 2-way set associative	16 KB, 1-way set associative	8 KB, 4-way set associative	32 KB, 32-way set associative	64 KB, 4-way set associative
Latency (clocks)	3	3	2	1	2
TLB entries (I/D/L2 TLB)	40/40/512/ 512	16	128/128	32/32	128/512
Minimum page size	4 KB	4 KB	4 KB	1 KB	8 KB
On-chip L2 cache	1024 KB, 16-way set associative	1024 KB, 4-way set associative	512 KB, 8-way set associative	—	—
Off-chip L2 cache	—	—	—	—	16 MB, 2-way set associative
Block size (L1/L2, bytes)	64	64	64/128	32	32

FIGURE 7.36 Desktop, embedded, and server microprocessors in 2004. From a memory hierarchy perspective, the primary differences between categories is the L2 cache. There is no L2 cache for the low-power embedded, a large on-chip L2 for the embedded and desktop, and 16 MB off chip for the server. The processor clock rates also vary: 0.4 GHz for low-power embedded, 1 GHz or higher for the rest. Note that UltraSPARC IV has two processors on the chip.

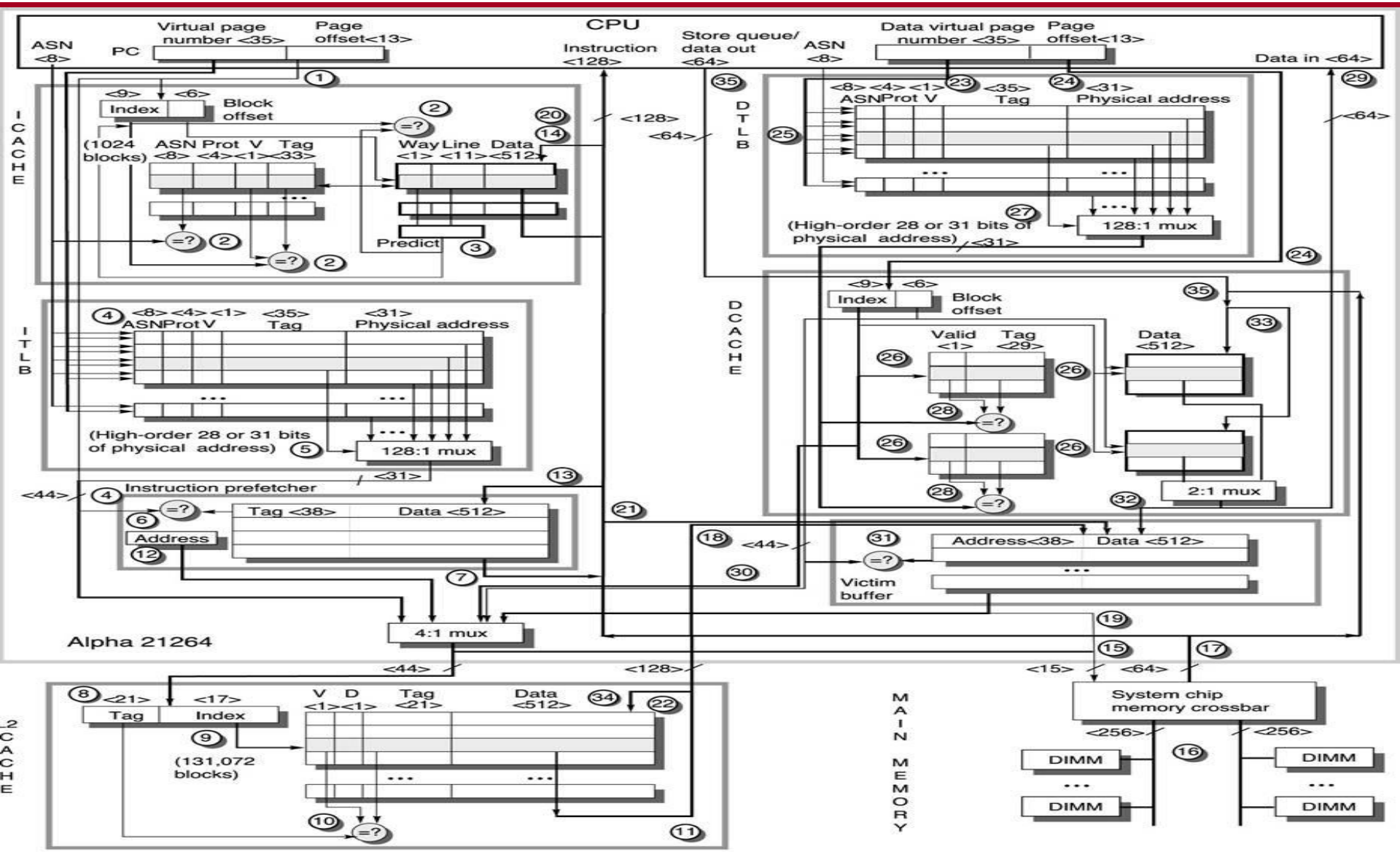
Η λειτουργία του Alpha AXP 21064 Data TLB κατά τη μετάφραση των διευθύνσεων



TLB = 32 blocks
Data cache = 256 blocks

Η πρόσβαση στα TLB γίνεται συνήθως με την τεχνική pipeline

Σύνοψη


CPU
ASN
PC
Virtual page number
Page offset
Instruction
Store queue/ data out
Data virtual page number
Data in
ICACHE
Index
Block offset
(1024 blocks)
ASN Prot V Tag
Way Line Data
Predict
DTLB
Physical address
128:1 mux
(High-order 28 or 31 bits of physical address)
ITLB
Instruction prefetcher
Tag
Data
Address
DCACHE
Valid
2:1 mux
Victim buffer
4:1 mux
Alpha 21264
L2 CACHE
(131,072 blocks)
MAIN MEMORY
System chip memory crossbar
DIMM

Read/Write (Intrinsity FastMATH cpu)