Αλγόριθμοι (πολιτικές) Αντικατάστασης Σελίδων

(Page Replacement Algorithms)

Οταν προκληθεί 'page-fault' (σφάλμα σελίδας) από μία αναφορά στη μνήμη κάποιας διεργασίας, η σελίδα που 'απουσιάζει' αναζητείται στο δίσκο και πρέπει να μεταφερθεί/τοποθετηθεί σε κάποιο ελεύθερο πλαίσιο (frame) της κύριας μνήμης (ταυτόχρονα πρέπει να επημερωθεί και ο πίνακας σελίδων της διεργασίας – page table)

Αν δεν υπάρχει ελύθερο πλαίσιο στη μνήμη πρέπει να επιλεγεί καποιο από τα 'γεμάτα' και να γίνει 'αντικατάσταση'. Τα περιεχόμενα (σελίδα) που βρίσκονταν στο συγκεκριμένο πλαίσιο (frame) μεταφέρονται στο δίσκο και το πλαίσιο πλέον εκχωρείται στη νέα σελίδα.

Αλγόριθμοι – Πολιτικές Αντικατάστασης

Επιλέγεται ένα από τα 'κατειλημένα'/γεμάτα πλαίσια της μνήμης (το οποίο ανήκει προφανώς σε κάποια διεργασία) σύμφωνα με κάποια κριτήρια όπως,

- Το χρόνο της CPU ή τον πραγματικό χρόνο που έχει περάσει από την τελευταία προσπέλαση της σελίδας που βρίσκεται στο πλαίσιο.
- Τη συχνότητα προσπέλασης της σελίδας
- Αν η διεργασία στην οποία ανήκει το πλαίσιο/σελίδα είναι κατάλληλη προς διανονή στην ΚΜΕ ή όχι
- Το ολικό πλήθος των τεμαχίων της διεργασίας αυτής, που βρίσκονται φορτωμένα στη μνήμη
- Την προτεραιότητα ή την ιεραρχία της διεργασίας αυτής σε σχέση με τη διεργασία που απαιτεί τη μνήμη
- Αν η σελίδα ανήκει σε μία μόνο διεργασία ή αν τη μοιράζονται πολλές διεργασίες.

Τοπικοί Αλγόριθμοι (Local Algorithms)

Επιλέγεται πλαίσιο/σελίδα προς αντικατάσταση, μόνο από αυτά που ανήκουν στην διεργασία η οποία προκάλεσε το 'page-fault'

Καθολικοί Αλγόριθμοι (Global Algorithms)

Μπορεί να επιλεγεί οποιοδήποτε πλαίσιο/σελίδα προς αντικατάσταση (ακόμα και αν ανήκει σε άλλη διεργασία)

Συγκεκριμένοι Αλγόριθμοι – Πολιτικές

Οι παρακάτω πολιτικές/αλγόριθμοι μπορούν να χρησιμοποιηθούν είτε τοπικά (local) είτε καθολικά (global)

Γενικός αρχικός σκοπός: να αντικαθιστούμε ένα πλαίσιο/σελίδα που να είναι εν γένει αρκετά 'αχρηστο' (να μην χρησιμοποιείται συχνά, να μην υπάρχει μεγάλη πιθανότητα να αναφερθεί/χρησιμοποιηθεί ξανά σύντομα).

1. Τυχαία Επιλογή

- Επιλέγεται τυχαία το πλαίσιο/σελίδα που θα αντικατασταθεί

2. FIFO (πρώτο μέσα – πρώτο έξω / first-in-first-out)

- Επιλέγεται το πλαίσιο/σελίδα που βρίσκεται στη μνήμη τον περισσότερο χρόνο (γηραιότερο)
- *Πρόβλημα:* η γηραιότερη σελίδα δεν είναι απαραίτητα η πιο άχρηστη

3. LRU (λιγότερο πρόσφατα χρησιμοποιημένο / least recently)

- Επιλέγεται το πλαίσιο/σελίδα που δεν έχει αναφερθεί για το μεγαλύτερο χρονικό διάστημα στο παρελθόν
- Υποθέτουμε ότι η μέλλουσα συμπεριφορά του προγράμματος θα είναι παρόμοια με την πρόσφατη συμπεριφορά του.
- Μειονέκτημα: Μεγάλο κόστος/χρόνος διαχείρισης
- Υλοποιείται συνήθως είτε μέσω 'υλικού' (μετρητές χρόνου κ.λ.π.) είτε με συνδυασμό 'υλικού' και 'λογισμικού' – ποτέ μόνο με 'λογισμικό' (για κάθε αναφορά στη μνήμη πρέπει να γίνει χρονοβόρα εργασία μορφής ταξινόμησης)

4. LFU (λιγότερο συχνά χρησιμοποιημένο / least frequently)

- Επιλέγεται το πλαίσιο/σελίδα που έχει χρησιμοποιηθεί λιγότερο συχνά κατά τη διάρκεια κάποιου προηγούμενου χρονικού διαστήματος.
- Παρόμοια πλεονεκτήματα/μειονεκτήματα με τον LRU. Τί γίνεται με τις σελίδες που μόλις φορτώθηκαν στη μνήμη??

Συγκεκριμένοι Αλγόριθμοι – Πολιτικές *(συνέχεια…)*

5. ***Second Chance (Αλγόριθμος Δεύτερης Ευκαιρίας)***

- Χρήση ‘used-bit’ – προσέγγιση LRU και LFU
- Εφαρμογή FIFO με τη διαφορά ότι: αν το επόμενο υποψήφιο ‘θύμα’ έχει ‘Used-bit’=1 του δίνουμε μία δεύτερη ευκαιρία (κάνουμε το ‘Used-bit’=0 και συνεχίζουμε το ψάξιμο)

6. ***Προτεραιότητες Πλαισίων/Σελίδων***

- Χρήση/εξέταση των ‘Uded-bit’ και ‘Dirty-bit’
- Η αναζήτηση για το υποψήφιο πλαίσιο/’θύμα’ γίνεται με σειρά προτεραιότητας με βάση τις ακόλουθες κατηγορίες:

  Τάξη 0 (0,0): εξετάζονται πρώτα τα πλαίσια/σελίδες εκείνα που ούτε έχουν ‘χρησιμοποιηθεί’ κατά το τελευταίο χρονικό διάστημα, ούτε έχουν ‘αλλαχθεί’

  Τάξη 1 (0,1): εξετάζονται μετά τα πλαίσια/σελίδες εκείνα που δεν έχουν ‘χρησιμοποιηθεί’ κατά το τελευταίο χρονικό διάστημα, αλλά έχουν ‘αλλαχθεί’

  Τάξη 2 (1,0): εξετάζονται στη συνέχεια τα πλαίσια/σελίδες εκείνα που ναι μεν έχουν ‘χρησιμοποιηθεί’ κατά το τελευταίο χρονικό διάστημα, αλλά δεν έχουν ‘αλλαχθεί’

  Τάξη 3 (1,1): τέλος εξετάζονται τα πλαίσια/σελίδες εκείνα που και έχουν ‘χρησιμοποιηθεί’ κατά το τελευταίο χρονικό διάστημα, και έχουν ‘αλλαχθεί’

7. ***Προτεραιότητες Διεργασιών***

- Επιλέγεται πάντα κάποιο πλαίσιο/σελίδα της διεργασίας με την μικρότερη προτεραιότητα

Σύνολο Εργασίας μίας Διεργασίας

(Working Set)

- Κατά τη διάρκεια της 'k' αναφοράς της διεργασίας στη μνήμη, ορίζεται ώς το σύνολο των πλαισίων/σελίδων που περιέχονται στην ακολουθία αναφοράς του προγράμματος της διεργασίας μέχρι την αναφορά 'k'.
- Η αλλοιώς: W(t,Δt): το σύνολο των πλαισίων που προσπελάστηκαν στο διάστημα (t-Δt, t)
- Δt → 'παράθυρο χρόνου'
- Μέγεθος Συνόλου Εργασίας W(t,Δt): ο πληθάριθμος του W(t,Δt):

Το Σύνολο Εργασίας για ένα καθορισμένο κατάλληλα διάστημα (παράθυρο χρόνου) Δt → θεωρούμε ότι είναι το απαραίτητο σύνολο πλαισίων/τεμαχίων που χρειάζεται η διεργασία ώστε να εκτελείται κανονικά/αποδοτικά (προκαλώντας λίγα page-faults). Θεωρούμε δε., ότι καλό είναι να βρίσκεται ολόκληρο στη μνήμη πάντοτε (ή το μεγαλύτερο μέρος του).

- ☞ Αν κάποια (ή κάποιες) διεργασίες δεν έχουν στη μνήμη ικανοποιητικό μέρος ή/και ολόκληρο το σύνολο εργασίας τους είναι πιθανόν να αρχίσουν να υπο-λειτουργούν (πολλά page-faults- μικρότερη πραγματική **εκτέλεση/utilization**)
- ☞ Αν πολλές διεργασίες αρχίσουν να συμπεριφέρονται κατ' αυτόν τον τρόπο, το σύστημα οδηγείται σε **λυγισμό/αλώνισμα (thrashing)**
- ☞ Για την αποφυγή του 'thrashing' το λειτουργικό σύστημα χρησιμοποιεί συνήθως διάφορα μέτρα που ελέγχουν το συνολικό 'utilization' και π.χ.
  - → αν διαπιστώσουν ότι 'πέφτει' κάτω από κάποιο όριο, αναστέλλουν κάποια ή κάποιες διεργασίες προσωρινά, δεν επιτρέπουν σε νέες διεργασίες να εισέλθουν (τις τοποθετούν σε ουρά) μέχρι να επανέλθει το utilization σε ανεκτό επίπεδο κ.λ.π.
- ☞ Η αρχή της τοπικότητας (locality)
  - → Κατά τη διάρκεια ενός χρονικού διαστήματος εκτέλεσης, μία διεργασία τίνει να χρησιμοποιεί ένα υποσύνολο των σελίδων της, και τα μέλη του συνόλου αυτού αλλάζουν σχετικά αργά.

ΑΣΚΗΣΕΙΣ

1. *(ανωμαλία Belady)*

Η ακολουθία αναφορών στη μνήμη ενός προγράμματος είναι :

0, 1, 2, 3, 0, 1, 4, 0, 1, 2, 3, 4

Πόσα 'page-faults' προκαλούνται για κάθε ένα από τους αλγορίθμους

→ FIFO
→ LRU

αν η μνήμη μας έχει
(α) 4 πλαίσια και
(β) 3 πλαίσια

2.

Η ακολουθία αναφορών στη μνήμη ενός προγράμματος είναι :

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0 ,1, …..
t0 t1 t2
Πόσα 'page-faults' προκαλούνται για κάθε ένα από τους αλγορίθμους

→ FIFO
→ LRU
→ LFU
→ Optimal *(επιλέγουμε το πλαίσιο που 'βλέπουμε' ότι θα ξαναζητηθεί αργότερα από όλα τα υπόλοιπα)*

αν η μνήμη μας έχει 3 πλαίσια

Να βρεθούν επίσης τα σύνολα εργασίας του προγράμματος στις χρονικές στιγμές t1 και t2 αν το παράθυρο χρόνου Δt είναι ίσο με 10

Αντικατάσταση Σελίδων - Ασκηση

- 1,3,5,4,2,7,5,6,2,8,2,5,3,6,8,2,4,1,3,6

Θεωρείστε ότι η παραπάνω ακολουθία αναφορών σελίδων παράγεται από μία (1) διαδικασία η οποία χρειάζεται συνολικά 8 σελίδες μνήμης και ξεκινάει να τρέχει σε ένα σύστημα με έξι (6) μόνο πλαίσια σελίδων (frames)
Με βάση την παραπάνω ακολουθία αναφορών και για κάθε έναν από τους αλγόριθμους αντικατάστασης σελίδων Fist In First Out (FIFO), Least Recently Used (LRU), Least Frequently Used (LFU), σας ζητείται να υπολογίσετε:

- σε κάθε χρονική στιγμή (ή για κάθε αναφορά σελίδας) ποιές σελίδες της διαδικασίας βρίσκονται στα πλαίσια σελίδων του συστήματος (πραγματοποιώντας αντικατάσταση σελίδων όποτε είναι απαραίτητο και σύμφωνα με τον εκάστοτε αλγόριθμο), συμπληρώνοντας κατάλληλα τους παρακάτω πίνακες αποτελεσμάτων, και
- τον συνολικό αριθμό σφαλμάτων σελίδας (page faults) τα οποία παράγονται συνολικά και καθ' όλη τη διάρκεια εκτέλεσης της διαδικασίας.

Λύση:

(α) Fisrt-In-First-Out (FIFO)

	1	3	5	4	2	7	5	6	2	8	2	5	3	6	8	2	4	1	3	6
Frame 1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
Frame 2		3	3	3	3	3	3	3	3	8	8	8	8	8	8	8	8	8	8	8
Frame 3			5	5	5	5	5	5	5	5	5	5	3	3	3	3	3	3	3	3
Frame 4				4	4	4	4	4	4	4	4	4	4	4	4	4	4	1	1	1
Frame 5					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Frame 6						7	7	7	7	7	7	7	7	7	7	7	7	7	7	7

Number of Page Faults: 10 (6 + 4)

(β) Least Recenly Used (LRU)

	1	3	5	4	2	7	5	6	2	8	2	5	3	6	8	2	4	1	3	6
Frame 1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	6	6	6
Frame 2		3	3	3	3	3	3	3	3	8	8	8	8	8	8	8	8	8	8	8
Frame 3			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	1	1	1
Frame 4				4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3
Frame 5					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Frame 6						7	7	7	7	7	7	7	7	7	7	7	4	4	4	4

Number of Page Faults: 11 (6 + 5)

(γ) Least Frequently Used (LFU)

	1	3	5	4	2	7	5	6	2	8	2	5	3	6	8	2	4	1	3	6
Frame 1	1	1	1	1	1	1	1	6	6	6	6	6	6	6	6	6	6	1	1	1
Frame 2		3	3	3	3	3	3	3	3	8	8	8	8	8	8	8	8	8	8	6
Frame 3			5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
Frame 4				4	4	4	4	4	4	4	4	4	3	3	3	3	3	3	3	3
Frame 5					2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
Frame 6						7	7	7	7	7	7	7	7	7	7	7	4	4	4	4

Number of Page Faults: 12 (6 + 6)

ΤΜΗΜΑΤΟΠΟΙΗΣΗ – ΤΕΜΑΧΙΣΜΟΣ (Segmentation)

Λογική οργάνωση του προγράμματος σε διάφορες δομοενότητες (modules) που μπορούν να μεταφραστούν, να προστατευτούν και να καταμεριστούν ανεξάρτητα η μία από την άλλη

Ο εικονικός (ιδεατός/λογικός) χώρος διευθύνσεων μίας διεργασίας διαιρείται σε (λογικά) τμήματα διαφορετικού μεγέθους, που αντιστοιχούν στις (λογικές) διαφορετικές δομοενότητες (modules) του προγράμματος

Κάθε λογική διεύθυνση 'U' (n bits) είναι στη μορφή:

Αριθμός Λογικού Τμήματος 's' (n-k bits)	*Μετατόπιση 'd' (k bits)*
	(αριθμός λέξης)

Μέγιστος αριθμός τμημάτων: 2^{n-k} Μέγιστο μέγεθος τμήματος: 2^k

Μετατρέπεται σε μία Φυσική Διεύθυνση 'Φ' στη μορφή

Διεύθυνση Βάσης Τμήματος	+	*Μετατόπιση 'd' (k bits)*

- Το πόσα bits είναι τελικά η φυσική διεύθυνση (m bits) εξαρτάται ουσιαστικά από το πραγματικό μέγεθος αυτής. Θα πρέπει να έχει τουλάχιστον 'k' bits (τουλάχιστον δηλαδή 2^k θέσεις – όσο και το μέγιστο μέγεθος ενός τμήματος). Συνήθως είναι 'n' bits (όσο και η λογική διεύθυνση - ώστε να μπορεί μία 'μέγιστη' σε μέγεθος με 2^{n-k} τμήματα διεργασία εν δυνάμει να χωράει ολόκληρη στη μνήμη ή – επίσης σύνηθες – μεγαλύτερη)

- Για κάθε διεργασία υπάρχει ένας Πίνακας Τμημάτων ο οποίος φορτώνεται σε κάποιο μέρος της μνήμης. Το σε ποια διεύθυνση της μνήμης αρχίζει ο πίνακας τμημάτων της διεργασίας μας το δείχνει ο καταχωρητής STBR (Segment Table Base Register)

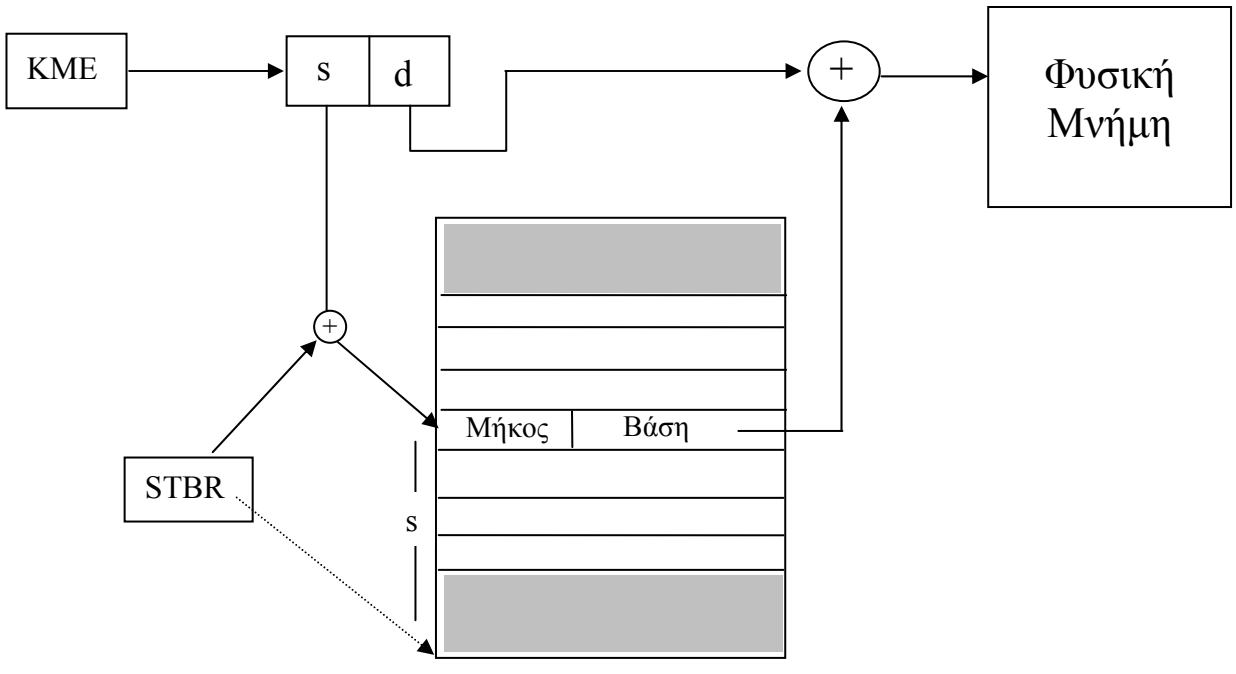
Segmentation – (…μετάβαση στη Φυσική Διεύθυνση…)

1. Το ‘s’ μας λέει σε ποιά θέση του πίνακα τμημάτων της διεργασίας να κοιτάξουμε

2. Από τη θέση αυτή του πίνακα τμημάτων λαμβάνουμε τη διεύθυνση ‘Βάσης’ η οποία δέιχνει σε ποιά διεύθυνση της ‘Φυσικής Μνήμης’ αρχίζει το συγκεκριμένο τμήμα, προσθέτουμε σε αυτήν το ‘d’ και έτσι εξάγουμε την τελική ‘Φυσική Διεύθυνση’ την οποία ψάχναμε.

3. Πρίν προστεθεί η μετατόπιση ‘d’ στη διεύθυνση ‘Βάσης’ του συγκεκριμένου τμήματος (που μας δείχνει ο πίνακας τμημάτων της διεργασίας), ελέγχεται αν το ‘d’ είναι μικρότερο από το ‘Μήκος’ του τμήματος (μας το δίνει επίσης η συγκεκριμένη θέση του πίνακα τμημάτων) και μόνο τότε προχωράμε (ειδάλλως: παραβίαση/σφάλμα)

4. Η πρόσβαση στον Πίνακα Τμημάτων (όπως και στη Σελιδοποίηση με τον Πίνακα Σελίδων) μας επιβαρύνει μία επιπλέον πρόσβαση στη μνήμη για κάθε αναφορά σε αυτήν. Για να έχουμε καλύτερη απόδοση, χρησιμοποιείται και εδώ η τεχνική των ‘συνειρμικών καταχωρητών’ (καταχωρητές συσχέτισης) όπου αποθηκεύονται οι πιο συχνά/πρόσφατα προσπελάσιμες εγγραφές του πίνακα τμημάτων της διεργασίας.



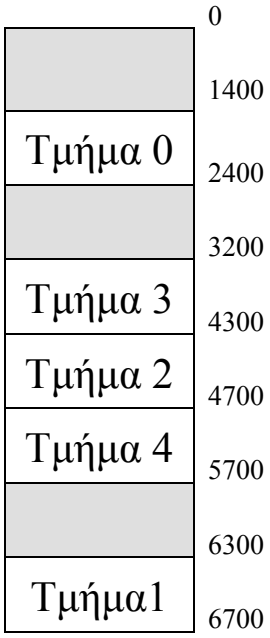
Πίνακας Τμημάτων

Segmentation – Ασκησεις/Παραδείγματα

Παράδειγμα: Αντιστοίχιση στη Φυσική Μνήμη (0 – 6700) ενός προγράμματος που έχει 5 τμήματα (οι αριθμοί στα δεξιά της Φυσικης μνήμης υποδηλώνουν σε ποιά διεύθυνση αρχίζει κάθε τμήμα).

Τμήμα	Μήκος	Βάση
0	1000	1400
1	400	6300
2	400	4300
3	1100	3200
4	1000	4700

Πίνακας Τμημάτων



Ασκηση

Υποθέστε ότι ο πίνακας Τμημάτων μίας διεργασίας περιέχει τα παρακάτω:

Τμήμα	Βάση	Μήκος
0	219	600
1	2300	14
2	90	100
3	1327	580
4	1952	96

Υπολογίστε ποιές είναι οι φυσικές διευθύνσεις που αντιστοιχούν στις εξής λογικές διευθύνσεις (s, d)

(a) 0, 430 (b) 1, 10 (c) 1, 11
(d) 2, 500 (e) 3, 400 (στ) 4, 112

Απαντήσεις:

(a) 649 (b) 2310 (c) 2311
(d) σφάλμα τμήματος (e) 1727 (στ) σφάλμα τμήματος

ΤΕΜΑΧΙΣΜΟΣ ΜΕ ΣΕΛΙΔΟΠΟΙΗΣΗ
(Paged Segmentation)

- Οταν τα τμήματα (segments) είναι πολύ μεγάλα…
- Μπορούμε να τα σελιδοποιήσουμε…
- Paged Segmentation = σελιδοποίηση για κάθε τμήμα
- Κάθε τμήμα μίας διεργασίας μπορεί να έχει (αποτελείται από) πολλές ‘σελίδες’ διασκορπισμένες στη Φυσική Μνήμη

<u>Η λογική διεύθυνση ‘U’ (n bits) τώρα χωρίζεται σε 3 τμήματα.</u>

Αριθμός Τμήματος ‘s’ (n-k bits)	*Μετατόπιση ‘d’ (k bits)*	
	Αρ. Σελίδας ‘p’	Μετατ. Σελίδας ‘d’’

<u>Και μετατρέπεται σε φυσική διεύθυνση ‘Φ’ στη μορφή:</u>

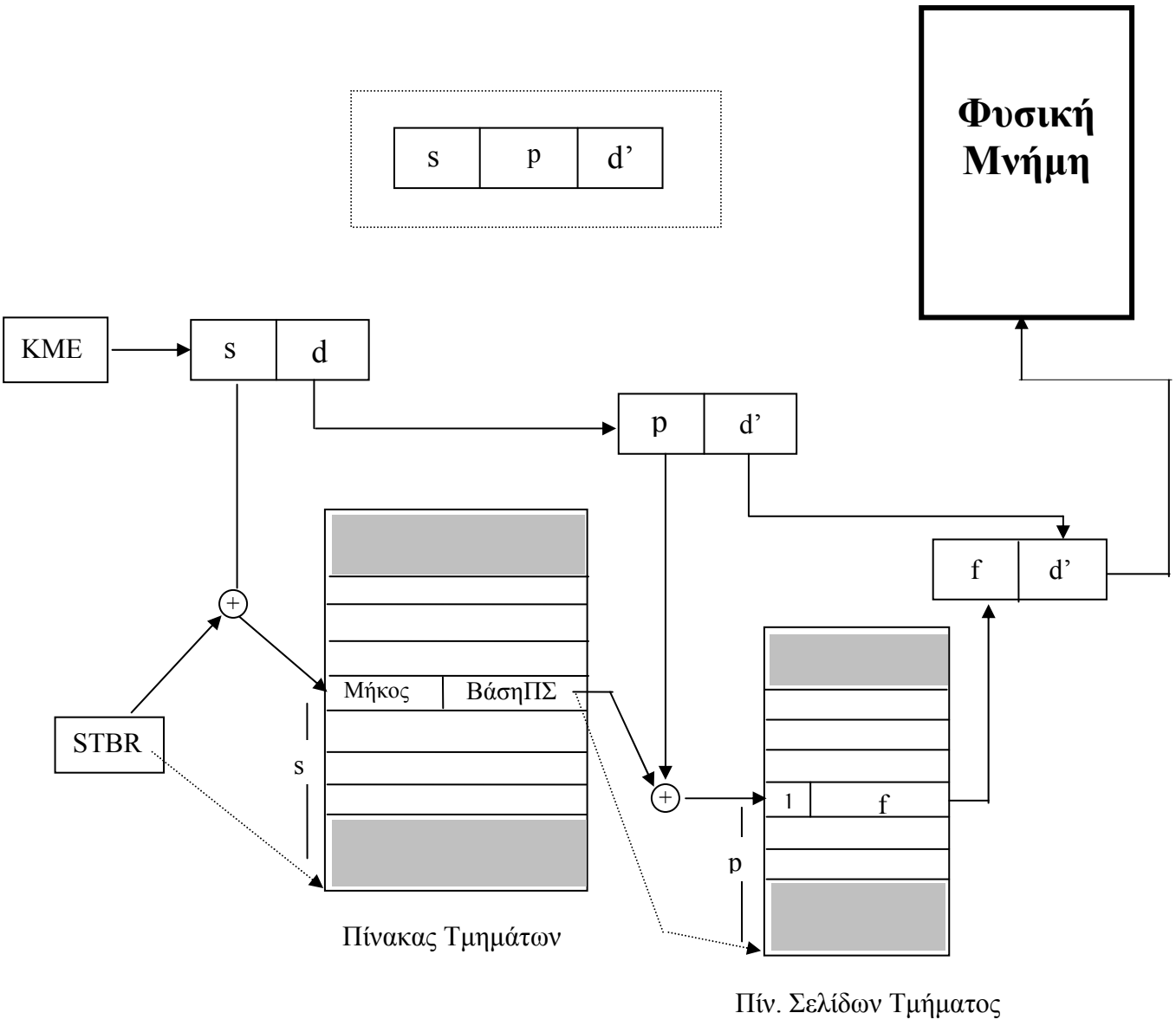
Αριθμός Φυσικής Σελίδας ‘f’ (*m-k’* bits)	Μετατόπιση ‘d’’ (*k’* bits)

- Οπου ‘m’ είναι τα bits της Τελικής Φυσικής Διεύθυνσης αν το συνολικό μέγεθος αυτής θεωρηθεί $M = 2^m$
- Υπάρχει ένας Πίνακας Σελίδων για κάθε Τμήμα
- Το ‘s’ (αριθμός τμήματος) μας δείχνει πάλι τη θέση του πίνακα τμημάτων που πρέπει να προσπελάσουμε. Από εκεί (αφού συγκρίνουμε το ‘d’ με το ‘Μήκος’ για να δούμε αν είναι μέσα στα επιτρεπόμενα όρια), λαμβάνουμε τη ‘Βάση’ η οποία τώρα είναι η Διεύθυνση Βάσης στη μνήμη από όπου αρχίζει ο Πίνακας Σελίδων του Τμήματος.
- Το ‘d’ (μετατόπιση) διαιρείται ακολούθως σε «αριθμό ιδεατής σελίδας» ‘p’ και «μετατόπιση σελίδας» ‘d’’ (όπως και στην απλή σελιδοποίηση). Το ‘p’ μας δείχνει σε ποιά θέση του Πίνακα Σελίδων του συγκεκριμένου Τμήματος να πάμε (προστίθεται για το λόγο αυτό στη ‘Βάση’ που μας δείχνει που αρχίζει ο Πίνακας Σελίδων του Τμήματος).

Paged Segmentation (…μετάβαση στη Φυσική Διεύθυνση…)

Από τη θέση αυτή (όπως στην απλή σελιδοποίηση) λαμβάνουμε (αν δούμε ότι το 'valid bit' είναι '1') τον «αριθμό» φυσικής σελίδας» 'f' τον οποίον στη συνέχεια συνδυάζουμε με τη «μετατόπιση σελίδας» 'd'' και καταλήγουμε έτσι στην τελική «Φυσική Διεύθυνση Μνήμης»

Τελικά είναι σαν όλος ο λογικός χώρος διευθύνσεων κάθε διεργασίας να ανάγεται πάλι σε 'σύνολο σελίδων' (σε επίπεδο φυσικής μνήμης). Η διαφορά είναι ότι σε 'λογικό επίπεδο' (επίπεδο λογικών διευθύνσεων/ αναφορών στο πρόγραμμα μίας διεργασίας) μπορούμε να χωρίζουμε και να αναφερόμαστε ξεχωριστά σε 'υποσύνολα σελίδων' του 'συνόλου σελίδων' της διεργασίας τα οποία είναι τα 'λογικά τμήματα' της διεργασίας.



Paged Segmentation – Ασκηση/Παράδειγμα

Θεωρείστε ένα σύστημα το οποίο για τη διαχείριση της μνήμης του χρησιμοποιεί τη μέθοδο της Σελιδοποιημένης Τμηματοποίησης (paged segmentation). Ο χώρος λογικών διευθύνσεων είναι 32-bits και διαχωρίζεται ως ακολούθως:

α) Ποιο είναι το μέγεθος σελίδας (page size) του συστήματος ;

β) Ποιο είναι το μέγιστο μέγεθος τμήματος (segment size) του συστήματος ; και από πόσες σελίδες (pages) μπορεί να αποτελείται ;

γ) Πόση είναι η εσωτερική κλασματοποίηση αν φορτώσουμε στο σύστημα αυτό ένα πρόγραμμα που αποτελείται από τα ακόλουθα τρία τμήματα: (i) ένα τμήμα των 166324 bytes για τον κώδικά του, (ii) ένα τμήμα των 16384 bytes για τις βιβλιοθήκες του και (iii) ένα τμήμα των 10240000 bytes για τα δεδομένα του ;

δ) Υποθέστε για ένα άλλο πρόγραμμα, ότι οι πίνακες τμημάτων και σελίδων έχουν την παρακάτω μορφή (σε δεκαεξαδική αναπαράσταση – πρέπει να τις μετατρέψετε σε δυαδικές):

Πίνακας Τμημάτων		Πίνακας Σελίδων Α		Πίνακας Σελίδων Β	
0	Διευθ.Πίν.Σελ.Α	0	CAFE	0	F000
1	Διευθ.Πίν.Σελ.Β	1	DEAD	1	D8BF
……	Τα υπόλοιπα 'invalid'	2	BEEF	……	Τα υπόλοιπα 'invalid'
……		3	BA11	……	

Ποια είναι η φυσική διεύθυνση που αντιστοιχεί σε κάθε μία από τις ακόλουθες (δεκαεξαδικές) λογικές διευθύνσεις: (i) '00000000', (ii) '20022002', (iii) '10015555', (iv) '01003342' ;

Λύση:

(α) Με 16 bits (offset) μπορούμε να διευθυνσιοδοτήσουμε 2^{16} διαφορετικές διευθύνσεις και άρα το μέγεθος της σελίδας είναι 2^{16} bytes = 2^6 K = 64K.

(β) Κάθε ένα από τα 2^4 τμήματα αποτελείται από 2^{12} σελίδες ή 4096 σελίδες. Αφού κάθε σελίδα έχει μέγεθος 64K το μέγιστο μέγεθος τμήματος είναι $2^{12}2^{16} = 2^{28}$ bytes ή 2^8 MB = 256 MB.

(γ) Το μέγεθος μιας σελίδας είναι 64 K, δηλαδή 65536 bytes. Επομένως:

Για την αποθήκευση του τμήματος του κώδικα απαιτούνται $\lceil 166324/65536 \rceil$ = 3 σελίδες, δηλαδή 3 * 64K = 192 K, ή 192 * 1024 bytes = 196608 bytes.

Για την αποθήκευση των βιβλιοθηκών απαιτείται $\lceil 16384/65536 \rceil$ = 1 σελίδα, δηλαδή 65536 bytes.

Τέλος, για την αποθήκευση των δεδομένων απαιτούνται $\lceil 10240000/65536 \rceil$ = 157 σελίδες, δηλαδή 10289152 bytes.

Η εσωτερική κλασματοποίηση είναι επομένως [(196608 – 166324) + (65536 – 16384) + (10289152 - 10240000)] bytes = (30284 + 49152+ 49152) bytes = 128588 bytes.

(δ) Υπολογισμός Φυσικών Διευθύνσεων

(i) Ο αριθμός $(00000000)_{16}$ μεταφράζεται στο δυαδικό σύστημα ως εξής:

0000 0000 0000 0000 0000 0000 0000 0000 ← δυαδικός

0 0 0 0 0 0 0 0 ← δεκαεξαδικός

Τα 4 πρώτα bits (0000) χρησιμοποιούνται για να βρούμε το τμήμα. Στην προκειμένη περίπτωση το τμήμα είναι το 0. Από το στοιχείο 0 του Πίνακα Τμημάτων βρίσκουμε την διεύθυνση του πίνακα σελίδων για αυτό το τμήμα, στην περίπτωση μας αυτός είναι ο πίνακας σελίδων Α και στη συνέχεια θα χρησιμοποιήσουμε τον πίνακα αυτό για να αναλύσουμε το υπόλοιπο της διεύθυνσης. Για να βρούμε αν η σελίδα βρίσκεται στη μνήμη εξετάζουμε τώρα τα επόμενα 12 bits της δυαδικής διεύθυνσης. Αφού είναι όλα 0, ψάχνουμε τη σελίδα 0 του τμήματος Α. Πληροφορίες για τη σελίδα αυτή υπάρχουν στο στοιχείο 0 του Πίνακα Σελίδων Α. Η σελίδα υπάρχει στη μνήμη και το πρώτο byte της βρίσκεται στη διεύθυνση $(CAFE)_{16}$. Η μετατόπιση μέσα στην σελίδα δίνεται από τα 16 λιγότερα σημαντικά (δεξιότερα) bits της διεύθυνσης (τα οποία θα αποτελέσουν επίσης τα 16 λιγότερα σημαντικά bits της φυσικής διεύθυνσης που αναζητούμε) και αφού όλα αυτά τα bits είναι 0, η φυσική διεύθυνση που αναζητούμε είναι η $(CAFE0000)_{16}$.

(ii) Η δεκαεξαδική διεύθυνση $(20022002)_{16}$ μεταφράζεται στην ακόλουθη διεύθυνση στο δυαδικό σύστημα:

0010 0000 0000 0010 0010 0000 0000 0010 ← δυαδικός

2 0 0 2 2 0 0 2 ← δεκαεξαδικός

Δουλεύουμε και πάλι όπως στο ερώτημα (i). Χρησιμοποιούμε τα 4 πρώτα bits για να βρούμε το τμήμα. Στην προκειμένη περίπτωση το τμήμα είναι το 2. Βλέπουμε από τον Πίνακα Τμημάτων ότι δεν υπάρχει έγκυρη εισαγωγή στο στοιχείο 2 του πίνακα και άρα η διεύθυνση αυτή δεν είναι έγκυρη.

(iii) Ομοίως, η δεκαεξαδική διεύθυνση $(10015555)_{16}$ μεταφράζεται στην ακόλουθη διεύθυνση στο δυαδικό σύστημα:

0001 0000 0000 0001 0101 0101 0101 0101 ← δυαδικός

1 0 0 1 5 5 5 5 ← δεκαεξαδικός

Τα 4 πρώτα bits (0001) χρησιμοποιούνται για να βρούμε το τμήμα. Στην προκειμένη περίπτωση το τμήμα είναι το 1. Από το στοιχείο 1 του Πίνακα Τμημάτων βρίσκουμε την διεύθυνση του πίνακα σελίδων για αυτό το τμήμα, στην περίπτωση μας αυτός είναι ο πίνακας σελίδων Β και στη συνέχεια θα χρησιμοποιήσουμε τον πίνακα αυτό για να αναλύσουμε το υπόλοιπο της διεύθυνσης. Για να βρούμε αν η σελίδα βρίσκεται στη μνήμη εξετάζουμε τώρα τα επόμενα 12 bits της δυαδικής διεύθυνσης. Τα bits αυτά (0000 0000 0001) μας πληροφορούν πως ψάχνουμε τη σελίδα 1 του τμήματος Β. Πληροφορίες για τη σελίδα αυτή υπάρχουν στο στοιχείο 1 του Πίνακα Σελίδων Β. Η σελίδα υπάρχει στη μνήμη και το πρώτο byte της βρίσκεται στη διεύθυνση $(D8BF)_{16}$. Η μετατόπιση μέσα στην σελίδα δίνεται από τα 16 λιγότερα σημαντικά (δεξιότερα) bits της διεύθυνσης (τα οποία θα αποτελέσουν επίσης τα 16 λιγότερα σημαντικά bits της φυσικής διεύθυνσης που αναζητούμε). Τα bits αυτά είναι 0101 0101 0101 0101, οπότε η φυσική διεύθυνση που αναζητούμε είναι η $(D8BF5555)_{16}$.

(iv) Τέλος, η δεκαεξαδική διεύθυνση $(01003342)_{16}$ μεταφράζεται στην ακόλουθη διεύθυνση στο δυαδικό σύστημα:

0000 0001 0000 0000 0011 0011 0100 0010 ← δυαδικός

0 1 0 0 3 3 4 2 ← δεκαεξαδικός

Χρησιμοποιούμε τα 4 πρώτα bits για να βρούμε το τμήμα. Στην προκειμένη περίπτωση το τμήμα είναι το 0. Από το στοιχείο 0 του Πίνακα Τμημάτων βρίσκουμε την διεύθυνση του πίνακα σελίδων για αυτό το τμήμα, στην περίπτωση μας αυτός είναι ο πίνακας σελίδων Α και στη συνέχεια θα χρησιμοποιήσουμε τον πίνακα αυτό για να αναλύσουμε το υπόλοιπο της διεύθυνσης. Για να βρούμε αν η σελίδα βρίσκεται στη μνήμη εξετάζουμε τώρα τα επόμενα 12 bits της δυαδικής διεύθυνσης. Τα bits αυτά (0001 0000 0000) μας πληροφορούν πως ψάχνουμε τη σελίδα $2^8 = 256$ του τμήματος Α.

Βλέπουμε από τον Πίνακα Σελίδων Α ότι δεν υπάρχει έγκυρη εισαγωγή στο στοιχείο 256 του πίνακα και άρα η διεύθυνση αυτή δεν είναι έγκυρη.