ΔΙΑΧΕΙΡΙΣΗ ΜΝΗΜΗΣ

Array N θέσεων (0….N-1 | $N=2^n$)

Ποιά τμήματα της μνήμης είναι σε χρήση και ποιά όχι, χορήγηση μνήμης σε διεργασίες όποτε τη χρειάζονται, εναλλαγή πληροφοριών (swapping) μεταξύ κύριας μνήμης και δίσκου κ.λ.π.

☞ Διαχωρισμός Συστημάτων διαχείρισης μνήμης

- ♦ Με εναλλαγή ή/και σελιδοποίηση (swapping, paging)
- ♦ Χωρίς τα παραπάνω – απλούστερα

Συστήματα Διαχείρισης Μνήμης χωρίς Εναλλαγή-Σελιδοποίηση

☞ ***Μονοπρογραμματισμός***

- ♦ Μία διεργασία χρησιμοποιεί όλη τη μνήμη (έως 1960 – δε χρησιμοποιείται πλέον – απαιτείται κάθε διεργασία να φέρει μαζί της και τους οδηγούς συσκευών που χρησιμοποιεί)
- ♦ Υπαρξη Λειτουργικού Συστήματος και Οδηγών Συσκευών σε ένα μέρος της μνήμης (ROM/RAM) και χρήση όλης της υπόλοιπης μνήμης από μία διεργασία

☞ ***Πολυπρογραμματισμός***

- ♦ Πολλές διεργασίες ταυτόχρονα στη μνήμη
- ♦ Ανάγκη: Πολλαπλές διεργασίες ανά εφαρμογή, πολλαπλοί χρήστες ταυτόχρονα (interactive), μεγαλύτερη αξιοποίηση της CPU (I/O και CPU bound)

Μοντελοποίηση Πολυπρογραμματισμού

1. *Εστω 5 διεργασίες με 20% του χρόνου τους για υπολογισμούς στη CPU*

2. *Εάν βρίσκονται ταυτόχρονα στη μνήμη έχουμε 100% αξιοποίηση της CPU ?*

3. *Οχι γιατί μπορεί να ζητούν ταυτόχρονα/επικαλυπτόμενα εξυπηρέτηση Εισόδου/Εξόδου*

Εστω:

'p' το ποσοστό χρόνου I/O κάθε διεργασίας και

'n' ο αριθμός των ταυτόχρονων διεργασιών στη μνήμη

('n': βαθμός πολυπρογραμματισμού)

Αξιοποίηση CPU = 1 - p^n

- Σχέση *μεγέθους μνήμης* (πόσες διεργασίες μπορούν να βρίσκονται στη μνήμη) και *αξιοποίησης CPU*

- Ανάλυση επιδόσεων με διαφορετικούς χρόνους άφιξης διεργασιών (και time sharing)

- Χρονοπρογραμματισμός Δύο Επιπέδων
  - επιλογή ταυτόχρονων διεργασιών στη μνήμη
  - αλγόριθμος χρονοπρογραμματισμού για αυτές

Σταθερά Τμήματα Μνήμης (Fixed Partitions)

Διαχωρισμός της μνήμης σε συγκεκριμένου μεγέθους τμήματα

☞ ***Αλγόριθμοι εκχώρησης τμημάτων σε διεργασίες***

- Πολλαπλές ουρές (μία για κάθε τμήμα)

  Πρόβλημα: γεμάτες και άδειες ουρές – αχρησιμοποίητα τμήματα

- Μία ουρά (επιλογή της πρώτης διεργασίας που χωράει)

  Πρόβλημα: σπατάλη μεγάλων τμημάτων για μικρές διεργασίες

- Μία ουρά (επιλογή της μεγαλύτερης διεργασίας που χωράει)

  Πρόβλημα: διάκριση εις βάρος των μικρών διεργασιών

  Λύσεις: Υπαρξη τουλάχιστον 1 μικρού τμήματος

  Μέγιστος αριθμός επιτρεπόμενων υπερπηδήσεων

☞ ***Μετατόπιση και Προστασία***

- Μετατόπιση κατά το φόρτωμα των διεργασιών
- Μετατόπιση κατά την εκτέλεση

Καταχωρητές βάσης και ορίου (base and limit registers)

Συστήματα με Εναλλαγή (swapping)

Ενα μέρος των διεργασιών μόνο βρίσκονται στη μνήμη – οι υπόλοιπες στο δίσκο – οποτεδήποτε μία διεργασία διακόπτεται μπορεί να μεταφερθεί στο δίσκο και να καταλάβει το χώρο της στη μνήμη μία άλλη διεργασία.

Μεταβλητά Τμήματα Μνήμης (variable partitions)

Δεν υπάρχουν συγκεκριμένα όρια και τμήματα – αυτά ορίζονται από το χώρο μνήμης που χρειάζεται κάθε διεργασία. Μεγαλύτερη ευελιξία έναντι των σταθερών τμημάτων, καλύτερη αξιοποίηση μνήμης, λιγότερη σπατάλη/αχρησιμοποίητα κομμάτια.

- ☞ Δημιουργούνται κενά τμήματα δυναμικά και τυχαία
- ☞ Συμπίεση μνήμης (compaction)
  - ♦ Ενοποίηση κενών και χρησιμοποιημένων τμημάτων μνήμης ώστε να υπάρχει ακόμα μεγαλύτερη ευελιξία και αξιοποίηση της μνήμης.
  - ♦ Κόστος: απαιτείται πολύς χρόνος από τη CPU για να γίνει συμπίεση – όσο συχνότερα γίνεται τόσο χειρότερα
  - ♦ Χρήση Ειδικού Υλικού για συμπίεση
- ☞ Απαίτηση επιπλέον μνήμης από μία διεργασία
  - ♦ Αν υπάρχει γειτονικός χώρος διαθέσιμος…. δίνεται
  - ♦ Αν όχι, μεταφέρεται σε άλλο κενό τμήμα (αν υπάρχει)
  - ♦ Ειδάλλως μεταφορά/εναλλαγή στο δίσκο
  - ♦ Αν δεν υπάρχει χώρος στο δίσκο …περιμένει ή καταστρέφεται…

Τρόποι Διαχείρισης/Παρακολούθησης Μεταβλητών Τμημάτων Μνήμης

☞ *Χάρτες Δυαδικών Ψηφίων (bitmaps)*

- Bit '0' ή '1' για κάθε στοιχειώδες τμήμα (block) μνήμης.
- Πολύ ικανοποιητικό σε χώρο – εξαρτάται από το μέγεθος του block (πόσος επιπλέον χώρος χριεάζεται για block 4 bytes) – πολύ εύκολη ενημέρωση σε μεταβολές
- Πρόβλημα: Αλγόριθμοι εκχώρησης - Τί γίνεται αν πρέπει να εκχωρηθεί σε μία διεργασία χώρος Κ συνεχόμενων blocks ? Μεγάλη καθυστέρηση αναζήτησης

☞ *Διαχείριση με Συνδεδεμένες Λίστες*

- Μία λίστα με όλα τα κενά και χρησιμοποιούμενα τμήματα της μνήμης σε σειρά διευθύνσεων
- Κάθε κόμβος/εγγραφή της λίστας περιέχει: (α) αν είναι κενό ή διεργασία (β) τη διεύθυνση αρχής (γ) το μέγεθός του (δ) δείκτη στον επόμενο κόμβο
- Πολύ εύκολη/άμεση ενημέρωση σε μεταβολές (με ενοποίηση κενών).
- Πολύ ευέλικτοι αλγόριθμοι εκχώρησης

☞ *Διαχείριση με το Σύστημα των Φίλων (Buddy System)*

- Στόχος: Επιτάχυνση συνένωσης κενών όταν έχουμε μεταβολές (τερματισμός διεργασίας, είσοδος νέας), με διατήρηση διαφορετικών λιστών κενών και αχρησιμοποίητων τμημάτων

Αλγόριθμοι Εκχώρησης Συνδεδεμένων Λιστών

- *First fit (πρώτης τοποθέτησης)*
  - Πολύ γρήγορος – δημιουργία κενών τυχαίου μεγέθους
- *Next fit (επόμενης τοποθέτησης)*
  - Παραλλαγή του first fit – λίγο χειρότερη απόδοση
- *Best fit (καλύτερης τοποθέτησης)*
  - Πιο αργός – δεν σπαταλάει/διασπά μεγάλα κενά ΑΛΛΑ δημιουργεί πολλά μικρά κενά
- *Worst fit (χειρότερης τοποθέτησης)*
  - …με σκοπό από τη διάσπαση τμημάτων να προκύπτουν κατά το δυνατόν μεγαλύτερα κενά
  - εξίσου αργός – εν γένει χειρότερος
- *Διατήρηση διαφορετικών λιστών κενών και χρησιμοποιούμενων τμημάτων - Λίστα κενών τμημάτων ταξινομημένη κατά μέγεθος*
  - Ο ‘Best fit’ γίνεται το ίδιο γρήγορος με τον ‘First fit’
  - Ο ‘Next fit’ δεν έχει λόγο ύπαρξης
  - Ο ‘Worst fit’ γίνεται ο γρηγορότερος
  - Προβλημα: καθυστέρηση ενημέρωσης σε μεταβολές
- *Αλγόριθμος ‘Quick Fit’ (γρήγορης τοποθέτησης)*
  - Τηρεί διαφορετικές λίστες για ορισμένα από τα περισσότερο συνηθισμένα μεγέθη μνήμης που ζητούνται
  - Πολύ γρήγορος/καθυστέρηση ενημέρωσης σε μεταβολές

ΠΑΡΑΔΕΙΓΜΑΤΑ - ΑΣΚΗΣΕΙΣ

(Variable Partitions)

Α. Εστω ένα σύστημα εναλλαγής μεταβλητών διαιρέσεων με τα ακόλουθα μεγέθη κενών κατά σειρά:

Α. 10K
Β: 4K
Γ: 20K
Δ: 18K
Ε: 7K
Ζ: 9K
Η: 12K
Θ: 15K

Εστω ότι καταφτάνουν αιτήσεις/διεργασίες με την εξής σειρά:

- 12K
- 10K
- 8K

Σε ποιό κενό μνήμης θα εκχωτηθεί κάθε μία από τις παραπάνω διεργασίες για κάθε έναν από τους αλγορίθμους
(i) first fit, (ii) best fit, (iii) worst fit και (iv) next fit ;

<u>Απάντηση:</u>

	First Fit	**Best Fit**	**Worst Fit**	**Next Fit**
12K	Γ	Η	Γ	Γ
10K	Α	Α	Δ	Δ
8K	Γ	Ζ	Θ	Ζ

Β. ***Υποθέστε ότι έχετε ένα σύστημα μνήμης μεταβλητών διαιρέσεων (χωρίς συμπίεση) το οποίο μία δεδομένη χρονική στιγμή αποκτά την ακόλουθη μορφή (έχει δηλαδή 5 διαθέσιμες 'οπές' για εκχώρηση μνήμης σε νέες διεργασίες, με αντίστοιχα μεγέθη 100Κ, 500Κ, 200Κ, 300Κ, 600Κ. Στη συνέχεια εμφανίζονται οι ακόλουθες κατά σειρά, διεργασίες αιτούμενες μνήμη για να εκτελεστούν: Α: 212Κ, Β: 417Κ, Γ: 112Κ, Δ: 426Κ.***

(α) Θεωρώντας πολιτική δρομολόγησης FCFS (First Come First Serve), δείξτε ποιο μέρος της διαθέσιμης μνήμης θα εκχωρηθεί σε κάθε διεργασία, για κάθε μία από τις ακόλουθες μεθόδους εκχώρησης:

1. πρώτου ταιριάσματος (first fit)
2. καλύτερου ταιριάσματος (best fit)

Είναι δυνατή η εκχώρηση μνήμης σε όλες τις διεργασίες χωρίς αναμονή, με κάθε μία από τις προαναφερόμενες μεθόδους εκχώρησης;

(β) Υποθέστε ότι στην ουρά των εισερχόμενων προς εκτέλεση εργασιών υπάρχει και μία πέμπτη διεργασία Ε:200Κ. Επαναλάβετε το ερώτημα (α) για τις ακόλουθες μεθόδους εκχώρησης:

1. καλύτερου ταιριάσματος (best fit)
2. υποβολής των διεργασιών σαν 'ομάδα' (batch) – χωρίς δηλαδή να παίζει ρόλο η σειρά άφιξής τους και διαλέγοντας εσείς την καλύτερη δυνατή κατανομή.

Συγκρίνετε τα αποτελέσματα που λαμβάνετε σε κάθε περίπτωση και εξηγείστε τα ποιοτικά, σε όχι περισσότερες από 3 γραμμές (γιατί δηλαδή συμβαίνει η μία να είναι καλύτερη της άλλης).

(γ) Υποθέστε τώρα ότι (έχοντας τις ίδιες 'οπές' διαθέσιμης μνήμης) εμφανίζονται οι ακόλουθες διεργασίες αιτούμενες μνήμη για να εκτελεστούν Α: 350Κ, Β: 200Κ, Γ:290Κ, Δ: 150Κ, Ε: 575Κ, ΣΤ: 50Κ. Βρείτε μία σειρά άφιξης των παραπάνω διεργασιών για την οποία η μέθοδος εκχώρησης 'πρώτου ταιριάσματος' (first fit) να δίνει καλύτερο αποτέλεσμα από την μέθοδο εκχώρησης 'καλύτερου ταιριάσματος' (best fit).

Γ. *Θεωρείστε ότι πέντε (5) διεργασίες P_1-P_5, αφίκνυνται σε ένα σύστημα μεταβλητών διαιρέσεων (variable partitions) με συνολική μνήμη 256K, κατά τους χρόνους άφιξης που υπαγορεύει ο πίνακας:*

Διαδικασία	*Χρόνος Άφιξης*	*Διάρκεια*	*Μνήμη*
P_1	0	3	100K
P_2	1	4	40K
P_3	3	2	120K
P_4	4	4	60K
P_5	6	3	140K

Στον παραπάνω πίνακα επίσης δίνονται (α) πόσες χρονικές στιγμές απαιτούνται για την ολοκλήρωση της κάθε διεργασίας (Διάρκεια) και (β) πόση μνήμη απαιτεί η κάθε διεργασία για την εκτέλεσή της.
Θεωρώντας ότι η μνήμη στην αρχή έχει την ακόλουθη μορφή:

και ότι για την δρομολόγηση των διεργασιών:

- *χρησιμοποιείται αλγόριθμος Round-Robin με κβάντο '1 χρονική στιγμή', όσον αφορά την εκτέλεσή πολλαπλών διεργασίων (που τους έχει διατεθεί η απαιτούμενη μνήμη) 'ταυτόχρονα' στη CPU.*

Σας ζητείται να βρείτε *(ξεκινώντας από τη χρονική στιγμή '1') πόσες χρονικές στιγμές απαιτούνται συνολικά για να ολοκληρωθεί η εκτέλεση όλων των διεργασιών (μέχρι να ελευθερώσει δηλαδή τη CPU και η τελευταία διεργασία),*

(α) χωρίς συμπίεση (compaction)
(β) με συμπίεση (compaction)

Για να τεκμηριώσετε τα αποτελέσματά σας φτιάξτε για κάθε περίπτωση (με και χωρίς συμπίεση), έναν πίνακα της ακόλουθης μορφής, ο οποίος θα απεικονίζει για κάθε χρονική στιγμή τόσο (α) ποια διεργασία βρίσκεται (εξυπηρετείται) στη CPU όσο και την τρέχουσα (κατά τη χρονική στιγμή εκείνη) 'εικόνα μνήμης':

Χρονική Στιγμή	*Αφιξη*	*Εικόνα Μνήμης*	*CPU*	*Ουρά Μνήμης*
0	P_1	---------	---------	P_1
1	P_2	<P_1-100> ……	P_1	P_2
2	……	……	……	……

ΙΔΕΑΤΗ ΜΝΗΜΗ (Virtual Memory)

- Οταν ένα μεγάλο πρόγραμμα δεν χωράει στη μνήμη…
- Οταν θέλουμε να βάζουμε στη μνήμη ταυτόχρονα κατά το δυνατόν περισσότερες διεργασίες ανεξαρτήτως του χώρου που απαιτούν
- Οταν θέλουμε να μην αφήνουμε καμία διεργασία που έρχεται στο σύστημα να περιμένει εκτός μνήμης

Ιδεατή Μνήμη (Virtual Memory)

Κάθε πρόγραμμα/διεργασία έχει ένα δικό του χώρο μνήμης/διευθύνσεων (ιδεατός χώρος διευθύνσεων) του οποίου ένα μέρος συνήθως (ή/και όλος στην ιδανική περίπτωση) βρίκεται στη φυσική μνήμη όταν τρέχει το πρόγραμμα

Σελιδοποίηση (Paging)

- ☞ Θεωρούμε τη μνήμη διαχωρισμένη σε ισομεγέθη τμήματα (πλαίσια σελίδας – page frames)
- ☞ Ο χώρος των διευθύνσεων κάθε προγράμματος/διεργασίας (ιδεατές διευθύνσεις) διαχωρίζονται επίσης σε ισομεγέθη τμήματα (σελίδες – pages)

Οταν χρησιμοποιείται 'paging' οι διευθύνσεις που παράγει ένα πρόγραμμα/διεργασία (ιδεατές διευθύνσεις) δεν οδεύουν κατευθείαν στην αρτηρία της μνήμης (memory bus)

Διεχετεύονται στη Μονάδα Διαχείρισης Μνήμης (MMU) η οποία τις αντιστοιχίζει σε διευθύνσεις της φυσικής μνήμης.

ΣΕΛΙΔΟΠΟΙΗΣΗ (συνέχεια)

Εστω Φυσική Μνήμη με (θέσεις/λέξεις/διευθύνσεις):

- Μέγεθος Φυσικής Μνήμης: $M = 2^m$ (π.χ. $64M = 2^{26}$)
- Μέγεθος Πλαισίου σελίδας = $P = 2^k$ (π.χ. $4K = 2^{12}$)
- Αριθμός Πλαισίων σελίδων = $M/P = 2^{m-k}$ (π.χ. $16K = 2^{14}$)

Εστω πρόγραμμα/διεργασία (ιδεατή μνήμη) με:

- Μέγεθος απαιτούμενης μνήμης $N = 2^n$ (π.χ. $2M = 2^{21}$)
- Μέγεθος Σελίδας = $P = 2^k$ (π.χ. $4K = 2^{12}$)
- Αριθμός Σελίδων = $N/P = 2^{n-k}$ (π.χ. $512 = 2^9$)

Η σημασία της 'δυαδικής λογικής'…

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

Αριθμός Ιδεατής Σελίδας 'p' (*n-k* bits)	Μετατόπιση 'd' (*k* bits)

Και μετατρέπεται σε φυσική διεύθυνση Φ (σύνολο m bits):

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

Τόσο ο 'ιδεατός χώρος διευθύνσεων' όσο και ο 'φυσικός χώρος διευθύνσεων' είναι 'ακολουθιακοί'

Εστω ιδεατή διεύθυνση προγράμματος *U* και μέγεθος σελίδας *P*

Τότε: $p = U \text{ div } P$ $d = U \bmod P$ | $U = (p \times P) + d$

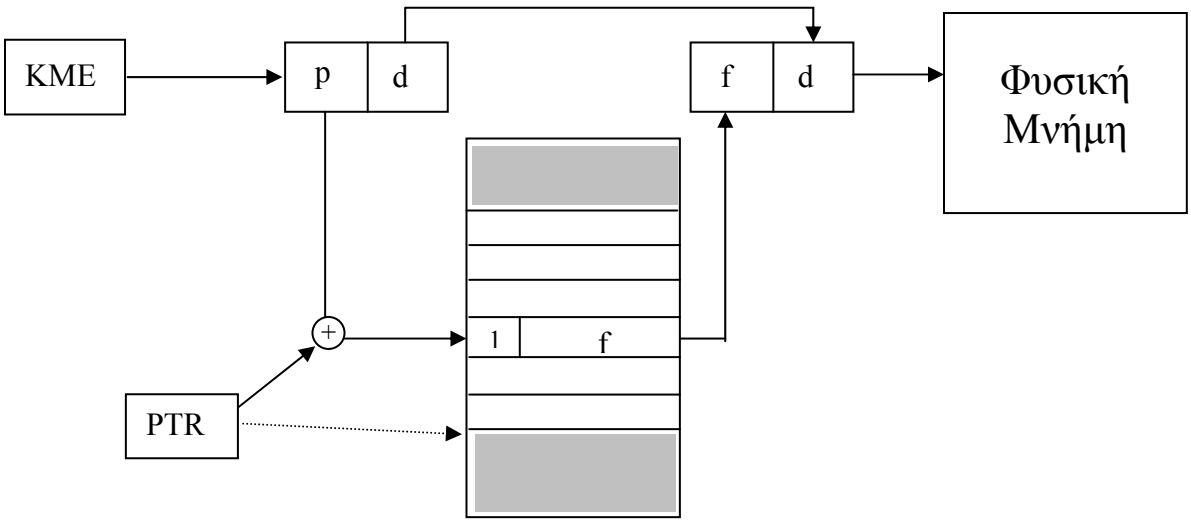
ΣΕΛΙΔΟΠΟΙΗΣΗ (συνέχεια)

Πίνακας Σελίδων:

Αντιστοιχίζει κάθε σελίδα του προγράμματος (ιδεατή μνήμη) σε πλαίσιο σελίδας της φυσικής μνήμης. Αρα έχει μήκος όσο και ο αριθμός ιδεατών σελίδων = N/P = 2^{n-k} (π.χ. 512 = 2^9)

Υπάρχει ένας καταχωρητής (Page Table Register – PTR) ο οποίος περιέχει τη διεύθυνση της φυσικής μνήμης όπου αρχίζει (έχει φορτωθεί) ο πίνακας σελίδων για την τρέχουσα διεργασία/πρόγραμμα

Τα πρώτα (n-k) bits της αρχικής (ιδεατής) διεύθυνσης (ο αριθμός δηλαδή ιδεατής σελίδας 'p') δείχνουν ποιά θέση του πίνακα σελίδων πρέπει να προσπελαστεί



Ελέγχεται το 'present bit' της συγκεκριμένης θέσης του πίνακα σελίδων για το αν η εν λόγω σελίδα βρίσκεται στη μνήμη ή όχι

Αν είναι '1' τότε λαμβάνεται το περιεχόμενο της θέσης αυτής του πίνακα σελίδων δηλαδή 'f'. Ο αριθμός της αρχικής ιδεατής σελίδας 'p' (*n-k* πρώτα bits της αρχικής ιδεατής διεύθυνσης) αντικαθίσταται από το 'f' (*m-k* bits / υψηλής τάξης bits) που υπαγορεύει ο πίνακας σελίδων και μαζί με το 'd' (*k* bits / χαμηλής τάξης bits) της αρχικής (ιδεατής διεύθυνσης) αποτελούν πλέον τη διεύθυνση της φυσικής μνήμης που πρέπει να προσπελαστεί.

ΣΕΛΙΔΟΠΟΙΗΣΗ (συνέχεια)

Αλλοιώς (γενικά και ανεξάρτητα με τη δυαδική λογική):

Τελική Διεύθυνση Φυσικής Μνήμης =
= 'Αριθμός Φυσικής Σελίδας' x 'Μέγεθος Σελίδας' + 'Μετατόπιση'

$f = \Phi \text{ div } P$ $d = \Phi \text{ mod } P$ | $\Phi = (f \times P) + d$

Αν το 'present bit' είναι '0' τότε η σελίδα δεν βρίσκεται στη φυσική μνήμη, δημιουργείται ένα 'σφάλμα σελίδας' ***(page fault) – trap*** *στο λειτουργικό σύστημα και πρέπει να φορτωθεί η σελίδα από το το δίσκο στη μνήμη (είτε σε κάποιο ελεύθερο μέρος/πλαίσιο ή αντικαθιστώντας κάποιο κατειλημένο.*

Με τη χρήση του Πίνακα Σελίδων απαιτούνται

→ 2 προσπελάσεις στη μνήμη για κάθε αναφορά

Για να βελτιωθεί (δραματικά) ο επιπλέον αυτός χρόνος Χρησιμοποιούνται:

→ Συσχετιστικοί (ή συνειρμικοί) καταχωρητές (associative registers)

Σε αυτούς αποθηκεύονται οι πιο συχνά προσπελάσιμες εγγραφές του πίνακα σελίδων της τρέχουσας διεργασίας. Ετσι, πρίν γίνει προσπέλαση του πίνακα σελίδων κάθε φορά στη μνήμη, ψάχνονται πρώτα οι εγγραφές αυτές (δραματικά γρηγορότερη η αναζήτηση αυτή) μήπως βρεθεί εκεί η απαιτούμενη εγγραφή εκεί. Με 16 τέτοιους καταχωρητές έχει αποδειχθεί ότι η πιθανότητα επιτυχίας είναι πολύ μεγάλη (αρκετή για να μην διαπιστώνεται σοβαρή καθυστέρηση)

ΠΑΡΑΔΕΙΓΜΑΤΑ - ΑΣΚΗΣΕΙΣ

Ιδεατή Μνήμη

0	0 1 2 3	A B C D
1	4 5 6 7	E F G H
2	8 9 10 11	I J K L
3	12 13 14 15	M N O P

Πίν. Σελ.

0	5
1	6
2	1
3	2

Φυσική Μνήμη

0	0 1 2 3	
1	4 5 6 7	I J K L
2	8 9 10 11	M N O P
3	12 13 14 15	
4	16 17 18 19	
5	20 21 22 23	A B C D
6	24 25 26 27	E F G H
7	28 29 30 31	

<u>Σελιδοποίηση για φυσική μνήμη 32 bytes με σελίδες των 4 bytes</u>

A. Εστω ότι τα 8 πλαίσια σελίδας της φυσικής μνήμης του συστήματος έχουν εκχωρηθεί στην τρέχουσα διεργασία ως ακολούθως:

Αριθμός πλαισίου φυσικής μνήμης	**Αριθμός σελίδας διεργασίας**
0	3
1	1
2	0
3	5
4	4
5	9
6	2
7	11

Υποθέστε ότι οι εικονικές διευθύνσεις έχουν μήκος 16 bits, εκ των οποίων 4 bits παριστάνουν τον αριθμό σελίδας και 12 bits τον ραιθμό της λέξης μέσα στη σελίδα (μετατόπιση).

Α. Κατασκευάστε τον πίνακα σελίδων της διεργασίας

Β. Βρείτε τις φυσικές διευθύνσεις στις οποίες αντιστοιχούν οι παρακάτω λογικές:

(α) 0 (β) 8192 (γ) 21500 (δ) 32780 (ε) 8196

Β. Ας υποθέσουμε ότι απαιτούνται 50 ns για το ψάξιμο των καταχωρητών συσχέτισης μίας συνειρμικής μνήμης ενός υπολογιστή που έχει χρόνο προσπέλασης μνήμης της τάξης των 750 ns. Να βρεθεί ο πραγματικός χρόνος προσπέλασης (effective access time) της μνήμης του υπολογιστή αυτού, όταν ο λόγος επιτυχίας (hit ratio) της συνειρμικής μνήμης είναι:

(α) 80% (β) 90%

Γ. Εστω ιδεατές διευθύνσεις των 8 bits και μέγεθος σελίδας PGSIZE

Αριθμός ιδεατής σελίδας --- (8-x) bits	Μετατόπιση --- (x) bits

$x=\log_2 PGSIZE$

Κάθε είσοδος του Πίνακα Σελίδων έχει την παρακάτω μορφή:

VALID_BIT	CHANGED_BIT	REF_BIT	(notused)	Αριθμός φυσικής σελίδας (10 bits)

Υποθέστε τώρα ότι, (α) η παράμετρος x είναι ίση με 3 (bits), (β) ότι η εικονική διεύθυνση αναφοράς στη μνήμη που παράγει κάποια χρονική στιγμή μία διαδικασία του συστήματος είναι «10101111», (γ) η τιμή του καταχωρητή PTR είναι ίση με '00000', (δ) πρόκειται να γίνει εγγραφή στη συγκεκριμένη θέση μνήμης και (ε) ο πίνακας σελίδων (page table) της διαδικασίας έχει τα παρακάτω περιεχόμενα:

00000	0	0	0	NU	1010000110
	0	0	0	NU	0010000110
	0	0	0	NU	0010001110
	0	0	0	NU	1010111110
	1	0	0	NU	0011100110
	1	1	0	NU	0011000110
	1	1	0	NU	0100000110
	1	1	0	NU	0000100111
	1	0	0	NU	1010001111
	1	0	0	NU	1010011110
	1	1	1	NU	1011110111
	1	1	1	NU	0010011111
	1	1	1	NU	1111000000
	1	1	1	NU	1110000100
	0	1	1	NU	0000000110
	0	1	1	NU	0000000010
	0	1	0	NU	0000001111
	0	1	0	NU	1111110110
	1	1	0	NU	1011111110
	1	1	0	NU	1010101010
	1	1	0	NU	0101010101
	1	0	0	NU	1110000111
	1	0	0	NU	1111001111
	1	0	0	NU	1100000011
	1	0	0	NU	0011111100
	1	1	1	NU	0001111000
	1	1	1	NU	0001110000
	1	1	1	NU	1111011111
	1	1	0	NU	0000100000
	1	1	0	NU	1111111100
	1	0	0	NU	0000101010
11111	1	0	0	NU	1010101111

Με βάση τα παραπάνω δεδομένα, (α) βρείτε ποιά γραμμή του πίνακα σελίδων πρέπει να χρησιμοποιηθεί για τη δεικτοδότηση στη φυσική μνήμη, (β) υπολογείστε βάση αυτής την τελική διεύθυνση αναφοράς στη μνήμη, και (γ) δώστε τη νέα μορφή της παραπάνω γραμμής του πίνακα σελίδων αν κάτι έχει διαφοροποιηθεί σε αυτήν.