ΠΕΡΙΕΧΟΜΕΝΑ

1 Θέματα Εξεταστικών

1.1 Φεβρουάριος και Σεπτέμβριος 2008

α)Οι διαθέσιμοι πόροι ενός συστήματος είναι: R1, R2, R3 (ένα αντίγραφο από κάθε πόρο). Επίσης υπάρχουν 3 διεργασίες οι A, B, C οι οποίες ζητούν (δεσμεύουν) και αποδεσμεύουν πόρους σύμφωνα με τον ακόλουθο τρόπο:

Process A	Process B	Process C
wants(R1)	wants(R2)	wants(R3)
wants(R2)	wants(R3)	wants(R1)
release(R1)	release(R2)	release(R3)
release(R2)	release(R3)	release(R1)

Αν το λειτουργικό ικανοποιήσει ακολουθιακά τις 3 διεργασίες με τη σειρά δηλαδή πρώτα την A, έπειτα την B και τέλος την C το σύστημα οδηγείται σε αδιέξοδο; Αν ναι που ακριβώς εμφανίζεται το αδιέξοδο, αν όχι ποιο το μειονέκτημα αυτού του τρόπου εκτέλεσης των διεργασιών;

Απάντηση

Η ακολουθιακή σειρά εκτέλεσης των διεργασιών δεν οδηγεί σε αδιέξοδο αφού δεν υπάρχει συναγωνισμός διεργασιών για πόρους. Αδιέξοδο συμβαίνει μόνο όταν εκτελούνται ταυτόχρονα πολλές διεργασίες οι οποίες ζητούν πόρους τους οποίους έχουν δεσμεύσει άλλες διεργασίες και δημιουργείται κυκλική αναμονή. Όταν εκτελείται μόνο μια διεργασία τότε παραβιάζεται η 4η συνθήκη (της κυκλικής αναμονής) από τις 4 αναγκαίες συνθήκες για εμφάνιση αδιεξόδου. Το μειονέκτημα της ακολουθιακής εκτέλεσης διεργασιών είναι ότι γίνεται υποχρησιμοποίηση πόρων διότι ενώ υπάρχουν 3 διαθέσιμοι πόροι χρησιμοποιούνται μόνο οι 2 από αυτούς και ένα δεύτερο μειονέκτημα είναι ότι καθυστερεί η εκτέλεση των διεργασιών.

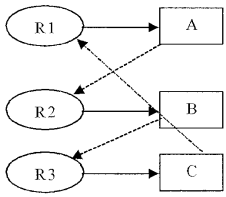
β)Οι διαθέσιμοι πόροι ενός συστήματος είναι πάλι: R1, R2, R3 (1 αντίγραφο από κάθε πόρο). Επίσης υπάρχουν πάλι 3 διεργασίες οι A, B, C οι οποίες ζητούν (δεσμεύουν) και αποδεσμεύουν πόρους σύμφωνα με τον ακόλουθο τρόπο:

Process A	wants(R1)
Process B	wants(R2)
Process C	wants(R3)
Process A	wants(R2)
Process B	wants(R3)
Process C	wants(R1)
Process A	release(R1)
Process A	release(R2)
Process B	release(R2)
Process B	release(R3)
Process C	release(R3)
Process C	release(R1)

Αν το λειτουργικό εκτελεί παράλληλα τις 3 διεργασίες το σύστημα οδηγείται σε αδιέξοδο; Αν ναι πως μπορεί αυτό να αντιμετωπιστεί;

Απάντηση

Δημιουργώντας το γράφο διεργασιών-πόρων παρατηρούμε ότι αυτό περιλαμβάνει κύκλο.



Άρα αυτό σημαίνει ότι το σύστημα οδηγείται σε αδιέξοδο. Για να αντιμετωπιστεί το αδιέξοδο πρέπει να εξυπηρετηθούν οι αιτήσεις με διαφορετική σειρά. Ένα πιθανό σενάριο που αποφεύγει το αδιέξοδο είναι το ακόλουθο:

Process A	Process B	Process C
wants(R1)	wants(R2)	
	wants(R3)	
	release(R2)	
wants(R2)	release(R3)	
release(R1)		wants(R3)
release(R2)		wants(R1)
		release(R3)
		release(R1)

1.2 Φεβρουάριος 2008

Ας υποθέσουμε ότι οι λογικές διευθύνσεις ενός προγράμματος αφορούν τις διευθύνσεις 0-8191 στις οποίες έχει φορτωθεί το λειτουργικό σύστημα. Γράψτε ένα ψευδοκώδικα που θα τοποθετεί τις (λογικές) διευθύνσεις του προγράμματος στις (φυσικές) διευθύνσεις μετά την 8191.

Απάντηση

```
for (i=0; i<8192; i++)
    mem[8192+i]=pr[i];
```

Ο πίνακας mem αντιπροσωπεύει την κύρια μνήμη που στις πρώτες θέσεις από 0-8191 έχει φορτωθεί το λειτουργικό (άρα μέγεθος φυσικού πλαισίου 8192 bytes), ενώ ο πίνακας program αντιπροσωπεύει τα περιεχόμενα μιας σελίδας που έχει βέβαια το ίδιο μέγεθος με το φυσικό πλαίσιο και σε αυτόν έχει φορτωθεί ένα πρόγραμμα χρήστη

1.3 Θέμα 2 Σεπτέμβριος 2009

Θέμα 2

Δίνονται οι ακόλουθες διεργασίες με τις προτεραιότητες και τη διάρκεια που φαίνεται στον πίνακα:

Διεργασία	Διάρκεια	Προτεραιότητα
Π1	1	5
Π2	2	4
Π3	2	3
Π4	5	1
Π5	15	2

Το κβάντο χρόνου (time slice) είναι 1 msec. Να εφαρμοστούν οι αλγόριθμοι

a)Round-Robin

b)Round-Robin με προτεραιότητες

c)Shortest Job First

και να υπολογιστούν για κάθε αλγόριθμο

α)Ο μέσος χρόνος απόκρισης

β)Ο μέσος χρόνος αναμονής

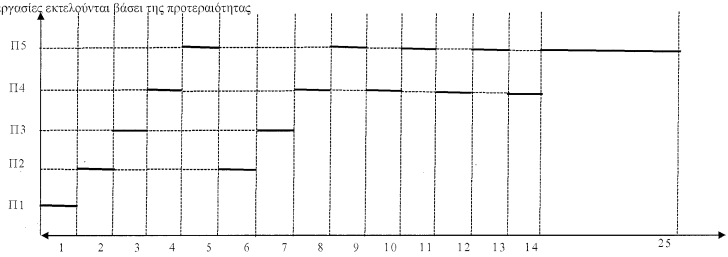
Απάντηση

a)Round-Robin

Οι διεργασίες εκτελούνται βάσει της σειράς άφιξης τους στην ουρά δηλαδή πρώτα εκτελείται η 1η διεργασία της ουράς μετά η δεύτερη

Ο χρόνος Απόκρισης κάθε διεργασίας είναι Χρόνος Αναμονής +Διάρκεια Εκτέλεσης

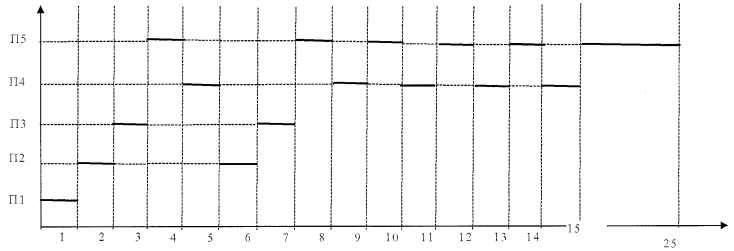
Οι διεργασίες εκτελούνται βάσει της προτεραιότητας



$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+4+5+9+10}{5} = 5{,}6 \quad \text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(4+2)+(5+2)+(9+5)+(10+15)}{5}$$

b) Round-Robin με προτεραιότητες

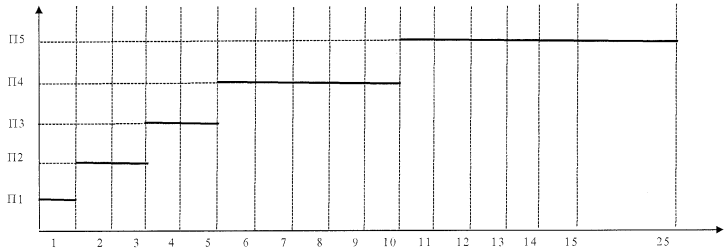
Οι διεργασίες εκτελούνται βάσει της προτεραιότητας τους και πιο συγκεκριμένα πρώτα τοποθετούνται στην ουρά οι διεργασίες με τη μεγαλύτερη προτεραιότητας και μετά οι διεργασίες με τη μικρότερη προτεραιότητα και με βάση τη σειρά αυτή εκτελούνται. Τώρα επειδή ο αλγόριθμος είναι συνδυασμός και Round Robin και προτεραιότητας, οι διεργασίες θα εκτελούνται για όσο χρόνο ορίζει το κβάντο και μετά θα διακόπτονται και εφόσον δεν έχουν ολοκληρωθεί θα τοποθετούνται πίσω στην ουρά.



$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+4+5+10+10}{5} = 5{,}8 \quad \text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(4+2)+(5+2)+(10+5)+(10+15)}{5}$$

c) SJF

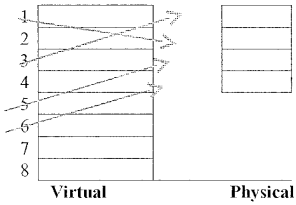
Οι διεργασίες εκτελούνται βάσει της διάρκειας τους και πιο συγκεκριμένα πρώτα τοποθετούνται στην ουρά οι διεργασίες με τη μικρότερη διάρκεια και μετά οι πιο χρονοβόρες διεργασίες και μετά εκτελούνται με βάσει αυτή τη σειρά. Εδώ δεν υπάρχει κβάντο χρόνου οπότε οι διεργασίες θα εκτελούνται πλήρως και δεν θα διακόπτονται.



$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+1+3+5+10}{5} = 3{,}8 \quad \text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(1+2)+(3+2)+(5+5)+(10+15)}{5}$$

Ο FIFO τις εξυπηρετεί με βάση την σειρά που φτάνουν στην ουρά και τις εξυπηρετεί πλήρως ως ότου να τελειώσουν (δεν υπάρχει κβάντο)

21. Θεωρείστε δύο διεργασίες Δ1 και Δ2 που τρέχουν σε ένα σύστημα με σελίδες/πλαίσια μεγέθους 8 KB και μέγεθος εικονικής μνήμης 64 KB. Το σχήμα απεικονίζει το χάρτη μνήμης (μέρος του ΠΣ) για τη Δ1



α. Μία αναφορά της Δ1 στην εικονική διεύθυνση 9000 θα προκαλέσει σφάλμα σελίδας → **Σωστό διότι η εικονική διεύθυνση 9000 ανήκει στη 2^n σελίδα και η 2^n σελίδα δεν έχει φορτωθεί σε φυσικό πλαίσιο μνήμης**

β. Η επόμενη αναφορά της Δ2 σε οποιαδήποτε εικονική διεύθυνση θα προκαλέσει σφάλμα σελίδας→ **Λάθος διότι οι διεργασίες μπορεί να χρησιμοποιούν κοινές διευθύνσεις με άλλες διεργασίες οπότε αν η αναφορά της Δ2 γίνει σε μια εικονική διεύθυνση της Δ1 η οποία απεικονίζεται σε φυσικό πλαίσιο μνήμης τότε δεν θα έχουμε σφάλμα σελίδας**

γ. Οι μισές αναφορές στη Δ1 θα προκαλέσουν σφάλμα→ **Λάθος, διότι μπορεί όλες οι αναφορές της Δ1 να γίνουν σε σελίδες που απεικονίζονται σε φυσικά πλαίσια και συνεπώς να μην έχουμε σφάλμα σελίδας**

δ. Όλες οι αναφορές στη Δ2 θα προκαλέσουν σφάλμα σελίδας→**Λάθος διότι μπορεί να γίνουν σε κοινές σελίδες με τη Δ1 που απεικονίζονται σε φυσικά πλαίσια**

ε. Αν η Δ1 διαγράψει τα περιεχόμενα της 1, 3, 5, 6 σελίδας, όλες οι μετέπειτα αναφορές της Δ1 σε αυτές τις σελίδες θα προκαλέσουν σφάλματα σελίδας→ **Σωστό, διότι δεν θα απεικονίζεται καμία σελίδα της Δ1 στη φυσική μνήμη**

Ανάπτυξης

1. Υποθέστε ότι ένα νήμα μιας εφαρμογής εκτελεί τον κώδικα «counter=counter+2» ενώ ένα δεύτερο νήμα εκτελεί ταυτόχρονα και ανεξάρτητα τον κώδικα «counter= counter*3». Η μεταβλητή «counter» είναι καθολική μεταβλητή του προγράμματος, έχει αρχική τιμή 4 και προσπελαύνεται μόνο από τις παραπάνω εντολές. Ποιό είναι το σύνολο των πιθανών τιμών του «counter» μετά το πέρας εκτέλεσης και των δύο νημάτων;

Απάντηση

Δεν μπορούν και τα δύο νήματα να κάνουν καταχώρηση counter είναι μια καθολική μεταβλητή στην οποία έχουν πρόσβαση και τα δύο νήματα. Θα πρέπει η μεταβλητή αυτή να προστατεύεται από μία μεταβλητή κλειδώματος (mutex). Γι αυτό ο κώδικας νημάτων θα τροποποιηθεί ως εξής:

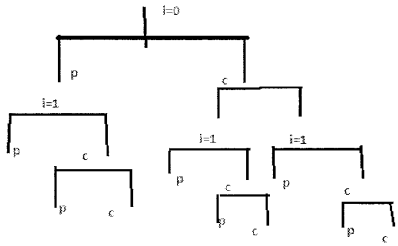
Νήμα 1	Νήμα 2
lock (mutex)	lock(mutex)
counter=counter+2	counter=counter*3
unlock (mutex);	unlock (mutex);

Αν το Νήμα 1 εκτελέσει πρώτα το lock (mutex) τότε η τιμή τής counter =6 και η τελική τιμή είναι counter=18
Αντίστοιχα για αν το Νήμα 2 εκτελέσει πρώτα το lock (mutex) η τιμή τής counter =12 και η τελική τιμή είναι counter=14.

2. Αν υποθέσουμε πως η μόνη διεργασία σε ένα σύστημα είναι αυτή που εκτελεί τον παρακάτω κώδικα, πόσες διεργασίες θα υπάρχουν συνολικά στο σύστημα μετά την εκτέλεση του κώδικα; Εξηγήστε ακριβώς πως καταλήγετε σε αυτό. Υποθέστε πως στο παράδειγμα μας η fork() επιτυγχάνει πάντοτε (Η fork() επιστρέφει 0 στη διεργασία παιδί και ένα αριθμό διάφορο του 0 στην διεργασία πατέρα)

```
for(i=0; i<2; i++) {
   id=fork()
   if (id==0)
      fork();
}
```

Απάντηση



3. Έστω ένα σύστημα με 16 bit εικονικές και φυσικές διευθύνσεις. Το μέγεθος κάθε σελίδας είναι 256 byte και έστω μία διεργασία η οποία περιέχει τα παρακάτω δεδομένα.

1	32	4F
1	1A	C3
1	89	22
0	42	82

Ο πίνακας περιέχει από αριστερά προς τα δεξιά το virtual bit, την εικονική σελίδα και το πλαίσιο σελίδας όλες σε 16δική μορφή.
α. Αν η διεργασία προσπελάσει την εικονική διεύθυνση 1AF2 ποιά θα είναι η φυσική διεύθυνση που θα τροποποιηθεί;
β. Αν η διεργασία προσπελάσει τη φυσική διεύθυνση 2222 ποιά θα είναι η εικονική διεύθυνση που θα τροποποιηθεί;
Απάντηση

1	32	4F
1	1A	C3
1	89	22
0	42	B2

α) 1AF2 εικονική διεύθυνση σε ποια φυσική διεύθυνση αντιστοιχεί;
Η εικονική διεύθυνση $(1AF2)_{16} = (6898)_{10}$. Ανήκει στη σελίδα 6898 div 256 = 26.94 ≈ 26. Η σελίδα 26 ξεκινάει από τη διεύθυνση 26x256=6656.Άρα το offset δηλαδή η μετατόπιση της διεύθυνσης (1AF2) αυτής από την αρχική διεύθυνση της σελίδας 26 είναι: 6898 – 6656 = 242 offset. Η εικονική σελίδα $(26)_{10} = (1A)_{16}$ αντιστοιχεί (απεικονίζεται) στο φυσικό πλαίσιο μνήμης $(C3)_{16} = (195)_{10}$. Το φυσικό πλαίσιο 195 αρχίζει από τη διεύθυνση 195x256 = 49920 + 242 offset = $(50162)_{10}$ και σε δεκαεξαδική απεικόνιση η φυσική διεύθυνση είναι C3F2.

Β τρόπος (για το α)
Αφού δίνεται η εικονική διεύθυνση 1AF2 και επειδή το μέγεθος σελίδας είναι 256 bytes (2^8=256) η διεύθυνση F2=$(242)_{10}$ ανήκει στη σελίδα 1A και αντιπροσωπεύει το offset. Η 1A απεικονίζεται στο C3 φυσικό πλαίσιο. Προσθέτουμε στο C3 το offset που είναι F2 και προκύπτει η φυσική διεύθυνση C3F2.

β) 2222 φυσική διεύθυνση σε ποια εικονική αντιστοιχεί;
Η φυσική διεύθυνση (2222)16 = (8738)10 ανήκει στο φυσικό πλαίσιο 8738 div 256 = 34 (ακέραιο πηλίκο). Το 34° φυσικό πλαίσιο ξεκινά από τη διεύθυνση 34x256= 8704 άρα το offset από την αρχή της σελίδας είναι:
8738-8704=34. Το φυσικό πλαίσιο $(34)_{10} = (22)_{16}$ αντιστοιχεί στην εικονική σελίδα $(89)_{16} = (137)_{10}$. Η σελίδα 137 αρχίζει από τη διεύθυνση 137x256=35072. Σε αυτή την αρχική διεύθυνση προσθέτουμε το offset που είναι 34 και προκύπτει η εικονική διεύθυνση που είναι 35072+34 =$(35106)_{10} = (8922)_{16}$.

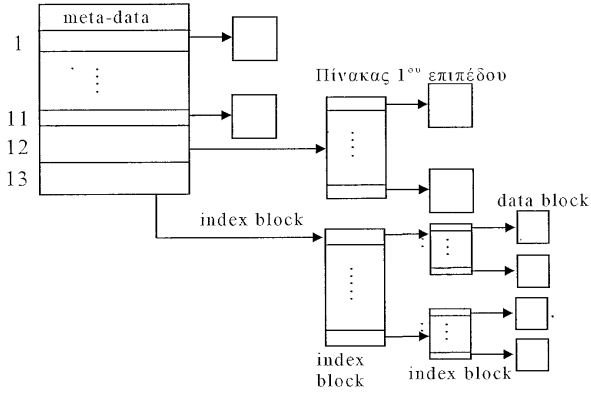
Β τρόπος (για το β)
Η φυσική διεύθυνση 2222 ανήκει στο φυσικό πλαίσιο σελίδας 22 καθώς το offset (22) είναι τα τελευταία 8 bit (τα τελευταία δύο δεκαεξαδικά ψηφία) λόγω του μεγέθους σελίδας που είναι 2^8 bytes. Επειδή $(22)_{16} = (34)_{10}$ είναι μικρότερο του 256 (μέγεθος σελίδας και φυσικού πλαισίου). Η φυσική σελίδα 22 αντιστοιχεί στην εικονική σελίδα 89, άρα προσθέτοντας το offset 22 στην σελίδα 89 προκύπτει 8922.

4. Υποθέστε ότι στη μνήμη μπορεί να βρίσκονται μέχρι 3 σελίδες. Μία διεργασία προσπελαύνει τις παρακάτω σελίδες:
3 0 4 2 3 2 3 8 4 3 2 1 2 0 1 7. Πόσα σφάλματα σελίδας θα συμβούν αν χρησιμοποιήσουμε τον αλγόριθμο ρολόι για την ακολουθία σελίδων δεδομένου ότι τα bit αναφοράς είναι 0.

5.Ένας σκληρός αποτελείται από 100 κυλίνδρους με τον κύλινδρο 0 να είναι πιο εξωτερικός .Η κεφαλή του δίσκου βρίσκεται αρχικά στον κύλινδρο 35 .Υπολογίστε το συνολικό μήκος μετακίνησης του βραχίονα σε κυλίνδρους για τους αλγόριθμους FIFO, SSF, SCAN όταν παρουσιάζονται οι ακόλουθες αιτήσεις 16 28 46 82 41 58 85 62 45 46

7. Θεωρείστε ένα file system με μέγεθος block 4.096 byte και 32 bit δείκτες προς το δίσκο και αρχεία. Κάθε αρχείο έχει 11 direct δείκτες προς block, 1 single indirect και 1 double indirect δείκτη.
α. Ποιο το μέγιστο μέγεθος αρχείου που υποστηρίζεται;
β. Ποιο το μέγιστο μέγεθος αρχείου που δε χρειάζεται double indirect δεικτοδότηση;
γ. Πόσες προσπελάσεις στο δίσκο θα χρειαστούμε για να διαβάσουμε το τελευταίο byte ενός αρχείου 512 MBytes υποθέτοντας ότι όλα τα δεδομένα που χρειάζεστε βρίσκονται στο δίσκο.

Απάντηση



α)Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 11 direct δείκτες είναι:
#direct pointers * block size = 11 * 4.096 = 45.056 bytes.

Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 single indirect δείκτη είναι:

$$\frac{4096}{4} * 4096 = 4.194.304 \text{ bytes}$$

Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 double indirect δείκτη είναι:

$$\left(\frac{4096}{4}\right)^2 * 4096 = 4.294.967.296 \text{ bytes}$$

Το μέγιστο μέγεθος αρχείου είναι: 4556 + 4.194.304 + 4.294.967.269 = 4.299.166.156 ≈ 4 GB

β) Το μέγιστο μέγεθος αρχείου χωρίς double – indirect δείκτη είναι:
45.056 + 4.194.304 ≈ 4 KB

γ) Το αρχείο μεγέθους 512 MB καταλαμβάνει συνολικά: 512 x 1024 x 1024: 4096 = 131.072 blocks

Τα πρώτα 11 blocks στα οποία δείχνουν οι 11 πρώτοι direct δείκτες θέλουν 1 προσπέλαση το καθένα. άρα συνολικά 11 προσπελάσεις. Μένουν 131.072 – 11 = 131.061 blocks.

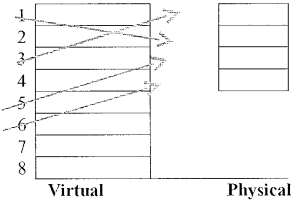
Ο single indirect δείκτης δείχνει σε 4096/4=1024 blocks. Άρα τα επόμενα 1024 blocks θέλουν 2 προσπελάσεις
Μένουν 131.061-1024=130.037 blocks

Ο double indirect δείκτης δείχνει σε $(4096/4)^2$=1.048.576 blocks. Άρα τα υπόλοιπα 130.037 blocks θέλουν 3 προσπελάσεις.

Για το τελευταίο μόνο byte του αρχείου απαιτούνται 3 προσπελάσεις.

1.4 Θέματα και Απαντήσεις Φεβρουάριος 2012

Β1. Θεωρείστε δύο διεργασίες Δ1 και Δ2 που τρέχουν σε ένα σύστημα με σελίδες/πλαίσια μεγέθους 8 KB και μέγεθος εικονικής μνήμης 64 KB. Το σχήμα απεικονίζει το χάρτη μνήμης (μέρος του ΠΣ) για τη Δ1. Απαντήστε στις επόμενες 3 ερωτήσεις



1. Μια αναφορά της Δ1 σε ποια από τις παρακάτω εικονικές διευθύνσεις θα προκαλέσει σφάλμα σελίδας (οι αριθμοί είναι στο δεκαδικό σύστημα)

α. 9.000

β. 16.500

γ.42.000

δ. Όλες οι παραπάνω

Απάντηση

Α/Α Σελίδας	Διευθύνσεις/Σελίδα	Presence Bit
1	0-8.191	**1**
2	8.192-16.383	**0**
3	16.384-24.575	**1**
4	24.576-32.767	**0**
5	32.768-40.959	**1**
6	40.960-49.151	**1**
7	49.152-57.343	**0**
8	57.344-65.535	**0**

Μία αναφορά της Δ1 στην εικονική διεύθυνση 9000 θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 9000 ανήκει στη 2η σελίδα και η 2η σελίδα δεν έχει φορτωθεί σε φυσικό πλαίσιο μνήμης**

Μία αναφορά της Δ1 στην εικονική διεύθυνση 16.500 ΔΕΝ θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 16.500 ανήκει στη 3η σελίδα και η 3η σελίδα έχει φορτωθεί ήδη σε φυσικό πλαίσιο μνήμης**

Μία αναφορά της Δ1 στην εικονική διεύθυνση 42.000 ΔΕΝ θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 42.000 ανήκει στη 6η σελίδα και η 6η σελίδα έχει φορτωθεί ήδη σε φυσικό πλαίσιο μνήμης**

Άρα η σωστή απάντηση είναι η α.

2. Η επόμενη αναφορά της Δ2 σε ποια από τις παρακάτω εικονικές διευθύνσεις θα προκαλέσει σφάλμα σελίδας (οι αριθμοί είναι στο δεκαδικό σύστημα)

α. 9.000

β. 16.500

γ.42.000

δ. Όλες οι παραπάνω

Απάντηση

δ. Όλες οι παραπάνω

Θεωρούμε ότι η Δ2 δεν έχει φορτώσει ακόμα στη φυσική μνήμη τις σελίδες που χρειάζεται οπότε θα προκληθεί σφάλμα σελίδας από όλες τις αναφορές της Δ2 στις προηγούμενες διευθύνσεις

3. Αν η Δ1 διαγράψει τα περιεχόμενα της 1ης 3ης 5η2 και 6ης σελίδας της, ποιες μετέπειτα αναφορές της Δ1 σε αυτές τις σελίδες θα προκαλέσουν σφάλματα σελίδας;

α. Αναφορές στις μονές σελίδες

β. Αναφορές στις ζυγές σελίδες

γ. Αναφορές σε όλες τις σελίδες

Απάντηση

γ. Αναφορές σε όλες τις σελίδες

Αν η Δ1 διαγράψει τα περιεχόμενα της 1, 3, 5, 6 σελίδας, όλες οι μετέπειτα αναφορές της Δ1 σε αυτές τις σελίδες θα προκαλέσουν σφάλματα σελίδας διότι δεν θα απεικονίζεται καμία σελίδα της Δ1 στη φυσική μνήμη

Β3. Υποθέστε ότι στη μνήμη μπορούν να βρίσκονται μέχρι 3 σελίδες. Μια διεργασία προελαύνει τις παρακάτω σελίδες μνήμης:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0

Πόσα page fault θα συμβούν αν χρησιμοποιήσουμε τον αλγόριθμο «Ρολόι» για αντικατάσταση σελίδων δεδομένου ότι όλα τα ψηφία R=0

Απάντηση

-	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
→	7*	7*	→7*	2*	2*	→2*	→2*	4*	4*	4*	→4	3*	3*	3	→3	0*	0*	0	0*
	→	0*	0*	→0	→0*	0	0*	→0	2*	2*	2	→2	→2*	1*	1*	→1*	→1*	7*	7*
		→	1*	1	1	3*	3*	3	→3	→3*	0*	0*	0*	→0	2*	2*	2*	→2	→2
	F	F	F	F	H	F	H	F	F	H	F	F	H	F	F	F	H	F	H

Άρα τα page faults είναι 13

Β4.Ένας σκληρός δίσκος αποτελείται από 100 κυλίνδρους με τον κύλινδρο 0 να είναι ο πλέον εξωτερικός. Η κεφαλή του δίσκου βρίσκεται αρχικά στον κύλινδρο 35. Υπολογίστε το συνολικό μήκος μετακίνησης του βραχίονα (σε κυλίνδρους) για τους αλγορίθμους FIFO, Shortest Seek Fit first και SCAN (Elevator) όταν παρουσιάζονται κατά σειρά οι παρακάτω αιτήσεις εξυπηρέτησης:
16, 28, 46, 82, 41, 58, 85, 62, 45, 46

Απάντηση

FIFO

16	**28**	**46**	**46**	**82**	**41**	**58**	**85**	**62**	**45**	
19	12	18	0	36	41	17	27	32	17	219

SSF

41	**45**	**46**	**46**	**58**	**62**	**82**	**85**	**28**	**16**	
6	4	1	0	12	4	20	35	57	12	151

SCAN

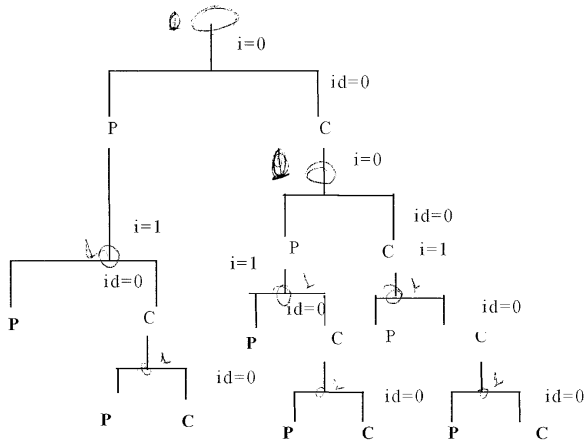
41	**45**	**46**	**46**	**58**	**62**	**82**	**85**	**99**	**28**	**16**	
6	4	1	0	12	4	20	35	14	71	12	179

Β5. Αν υποθέσουμε πως η μόνη διεργασία σε ένα σύστημα είναι αυτή που εκτελεί τον παρακάτω κώδικα, πόσες διεργασίες θα υπάρχουν συνολικά στο σύστημα μετά την εκτέλεση του κώδικα; Εξηγήστε ακριβώς πως καταλήγετε σε αυτό. Υποθέστε πως στο παράδειγμα μας η fork() επιτυγχάνει πάντοτε (Η fork() επιστρέφει 0 στη διεργασία παιδί και ένα αριθμό διάφορο του 0 στην διεργασία πατέρα)

```
for(i=0; i<2; i++) {
    id=fork();

    if (id==0) {
        fork();
    }
}
```

Απάντηση



Τελικά θα έχουμε 9 διεργασίες (όσα και τα φύλλα του τελικού δέντρου)

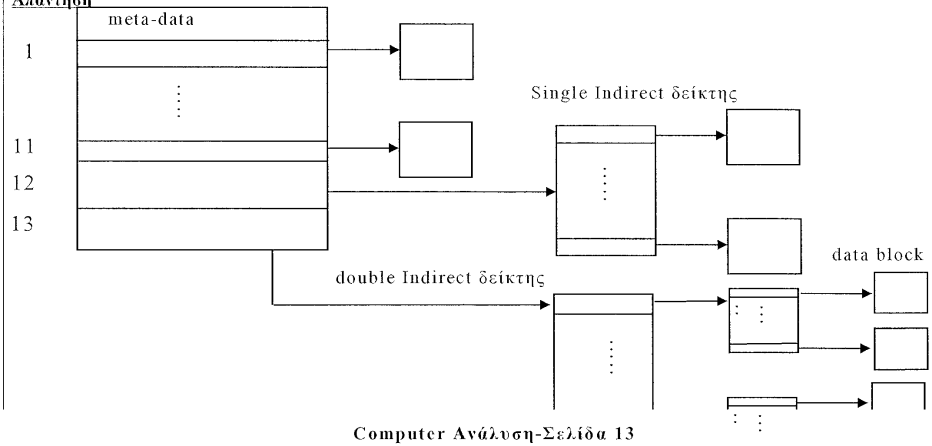
Β7. Θεωρείστε ένα file system με μέγεθος block 4.096 bytes και 32-bit δείκτες προς το δίσκο και αρχεία. Κάθε αρχείο έχει 11 direct δείκτες προς block, 1 single indirect δείκτη και 1 double indirect δείκτη.

a) Ποιο το μέγιστο μέγεθος αρχείου που υποστηρίζεται; Ποιο είναι το μέγιστο μέγεθος αρχείου που δεν χρειάζεται double indirect δείκτη;

b) Πόσες προσπελάσεις θα χρειαστούμε στο δίσκο για να διαβάσουμε το τελευταίο byte ενός αρχείου 1 GB (υποθέστε ότι όλα τα δεδομένα που χρειάζεστε βρίσκονται αρχικά στο δίσκο);

c) Έστω ότι έχετε ένα σκληρό δίσκο μεγέθους 1 TB και sectors 4KB και θέλετε να τον αξιοποιήσετε σε ένα σύστημα που το file system του χρησιμοποιεί δείκτες σε block μεγέθους 16-bit. Μπορείτε να βγάλετε συμπέρασμα αν θα δουλέψει/αναγνωριστεί ο δίσκος;

Απάντηση



a)Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 11 direct δείκτες είναι:
#direct pointers * block size = 11 * 4096 = 4556 bytes.

Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 single indirect δείκτη είναι:

$$\frac{4096}{4} * 4096 = 4.194.304 \text{ bytes}$$

Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 double indirect δείκτη είναι:

$$\left(\frac{4096}{4}\right)^2 * 4096 = 4.294.967.296 \text{ bytes}$$

Το μέγιστο μέγεθος αρχείου είναι: 4.556 + 4.194.304 + 4.294.967.269 = 4.299.166.156 ≈ 4 GB

Το μέγιστο μέγεθος αρχείου χωρίς double – indirect δείκτη είναι: 4.556 + 4.194.304 ≈ 4 KB

b) Το αρχείο μεγέθους 1GB=1024 MB=1024 x 1024KB = 1024 x 1024 x 1024 bytes καταλαμβάνει συνολικά: 1024 x 1024 x 1024: 4096 = 262.144 blocks
Τα πρώτα 11 blocks στα οποία δείχνουν οι 11 πρώτοι direct δείκτες θέλουν 1 προσπέλαση το καθένα, άρα συνολικά 11 προσπελάσεις. Μένουν 262.144 – 11 = 262.133 blocks.

Ο single indirect δείκτης δείχνει σε 4.096/4=1.024 blocks. Άρα τα επόμενα 1024 blocks θέλουν 2 προσπελάσεις
Μένουν 262.133-1024=261.109 blocks

Ο double indirect δείκτης δείχνει σε $(4096/4)^2$=1.048.576 blocks. Άρα τα υπόλοιπα 261.109 blocks θέλουν 3 προσπελάσεις.

Για το τελευταίο μόνο byte του αρχείου απαιτούνται 3 προσπελάσεις.

c) Το μέγεθος του δίσκου είναι Αριθμός κυλίνδρων x Αριθμός Sectors/κύλινδρο x Αριθμός μπλοκ/sector x bytes/μπλοκ

Αφού δεν δίνεται στην εκφώνηση ο αριθμός κυλίνδρων και ο αριθμός των Sectors/κύλινδρο δεν μπορούμε να βγάλουμε συμπέρασμα για το αν θα δουλέψει ο δίσκος.

2 Προτεινόμενα Θέματα Ασκήσεων Εξεταστικών

2.1

2.2 Ασκήσεις με CPU-Utilization

2.2.1 Άσκηση 1

Ένα υπολογιστικό σύστημα έχει αρκετό χώρο για να διατηρήσει τέσσερα προγράμματα στην κύρια μνήμη του. Τα προγράμματα αυτά μένουν αδρανή αναμένοντας είσοδο/έξοδο (Ε/Ε) το μισό χρόνο. Ποιο ποσοστό του χρόνου της CPU σπαταλιέται;

Answer

Η αναμονή για Ε/Ε είναι 50%, άρα p=0.5. Οι διεργασίες είναι τέσσερις οπότε n=4.

Βαθμός Χρήσης CPU = $1 - p^n = 1 - 0.5^4 = 0,9375$ ή 93,75%

Το ποσοστό του χρόνου της CPU σπαταλιέται είναι: 100 - 93,75 = 6,25%

2.2.2 Άσκηση 2

Υποθέτουμε ότι ένας υπολογιστής έχει 2GBytes μνήμης από τα οποία το λειτουργικό σύστημα χρησιμοποιεί το 1GB και κάθε πρόγραμμα χρήστη χρησιμοποιεί 512MB. Αν όλα τα προγράμματα έχουν αναμονή λόγω Ε/Ε, 60%

1. **ποιος είναι ο βαθμός χρήσης της CPU;**
2. **σε τι ποσοστό θα ανέβει η βαθμός χρήσης να προσθέσουμε 1GB ακόμη;**

Απάντηση

Διαθέσιμο είναι το 1GB. Αφού το κάθε πρόγραμμα χρησιμοποιεί 512MB μπορούν να υπάρχουν μέχρι 2 προγράμματα ταυτόχρονα στη μνήμη (n=2). Η αναμονή για Ε/Ε είναι 60%, δηλαδή p=0.6.

Βαθμός Χρήσης CPU = $1 - p^n = 1 - 0.6^2 = 0.64$ (64%)

Με 1GB ακόμη στη μνήμη μπορούμε να έχουμε συνολικά επιπλέον 2 προγράμματα, έτσι n=4

Βαθμός Χρήσης CPU = $1 - p^n = 1 - 0.6^4 = 0.8704$ (87,04%)

Ο βαθμός χρήσης έχει ανέβει σε ποσοστό (0.8704 - 0.64)/0.64 = 0,36 (36%)

2.3 Ασκήσεις με Χρονοπρογραμματισμό CPU

2.3.1 Άσκηση 1

5.7 Θεωρήστε ένα σύστημα που τρέχει δέκα έργα εξαρτώμενα από Ε/Ε και ένα έργο εξαρτώμενο από την ΚΜΕ. Υποθέστε ότι τα έργα που είναι εξαρτώμενα από Ε/Ε ζητούν μία λειτουργία Ε/Ε για κάθε ms υπολογισμού της ΚΜΕ, και ότι κάθε λειτουργία Ε/Ε απαιτεί 10 ms για την ολοκλήρωσή της. Υποθέστε επίσης ότι η επιβάρυνση για την εναλλαγή περιβάλλοντος λειτουργίας είναι 0.1 ms και ότι όλες οι διεργασίες είναι μεγάλης διάρκειας έργα. Ποια είναι η χρησιμοποίηση της ΚΜΕ για ένα χρονοπρογραμματιστή εκ περιτροπής όταν:
α. το κβάντο χρόνου είναι 1 ms
β. το κβάντο χρόνου είναι 10 ms

Απάντηση
α. Ο πρώτος κύκλος των 11 συνολικά διεργασιών ολοκληρώνεται μετά από (11*1)+(11*0,1)=12,1 ms. Τη χρονική στιγμή που ολοκληρώνεται ο πρώτος κύκλος των διεργασιών έχει λήξει η Ε/Ε που ζήτησε η $1^η$ διεργασία και άρα η πρώτη διεργασία μπορεί αμέσως να ξεκινήσει το νέο κύκλο. Όταν μετά από 1 ms η διεργασία αυτή ζητήσει τη νέα Ε/Ε, θα αναλάβει την ΚΜΕ η επόμενη (αφού έχει λήξει η Ε/Ε που ζήτησε), κ.ο.κ. Έτσι λοιπόν στα 12,1 ms της διάρκειας ενός κύκλου, τα 11 είναι χρήσιμα για το σύστημα και το 1 χαμένο. Άρα η χρησιμοποίηση της ΚΜΕ είναι 100*(11/12,1)=90,9 δηλαδή 90,9%.

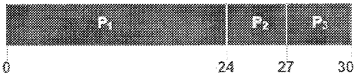
β. Ο πρώτος κύκλος των 11 συνολικά διεργασιών ολοκληρώνεται μετά από (10*1)+10+(11*0,1)=21,1 ms. Τη χρονική στιγμή που ολοκληρώνεται ο πρώτος κύκλος των διεργασιών έχει λήξει η Ε/Ε που ζήτησε η $1^η$ διεργασία και άρα η πρώτη διεργασία μπορεί αμέσως να ξεκινήσει το νέο κύκλο. Όταν μετά από 1 ms η διεργασία αυτή ζητήσει τη νέα Ε/Ε, θα αναλάβει την ΚΜΕ η επόμενη (αφού έχει λήξει η Ε/Ε που ζήτησε), κ.ο.κ. Έτσι λοιπόν στα 21,1 ms της διάρκειας ενός κύκλου, τα 20 είναι χρήσιμα για το σύστημα και το 1,1 χαμένο. Άρα η χρησιμοποίηση της ΚΜΕ είναι 100*(20/21,1)=94,7, δηλαδή 94,7%.

2.3.2 Άσκηση 2 με χρόνους Απόκρισης και Αναμονής

Αλγόριθμος χρονοδρομολόγησης με σειρά άφιξης (First Come First Served)

Διεργασία	Χρόνος έκρηξης
P_1	24
P_2	3
P_3	3

- Υποθέτουμε ότι οι διεργασίες φτάνουν με τη σειρά: P_1, P_2, P_3
  Το διάγραμμα Gantt για τη δρομολόγηση είναι:

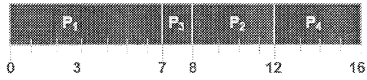


- Χρόνος αναμονής για P_1 = 0, για P_2 = 24, για P_3 = 27
- Μέσος χρόνος αναμονής: (0 + 24 + 27)/3 = 17
- Μέσος χρόνος απόκρισης: (0+24 + 24+3 +27+3)/3=27

Παράδειγμα μη-προεκτοπιστικού SJF

Διεργασία	Χρόνος άφιξης	Χρόνος έκρηξης
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

- SJF (μη προεκτοπιστικός)

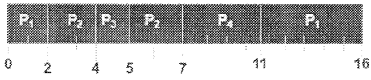


- Μέσος χρόνος αναμονής = (0 + 6 + 3 + 7)/4 = 4
- Μέσος χρόνος απόκρισης = (0+7 + 6+4 + 3+1 + 7+4)/4 = 8

Παράδειγμα προεκτοπιστικού SJF

Διεργασία	Χρόνος άφιξης	Χρόνος έκρηξης
P1	0.0	7
P2	2.0	4
P3	4.0	1
P4	5.0	4

- SJF (προεκτοπιστικός)



- Μέσος χρόνος αναμονής = (9 + 1 + 0 + 2)/4 = 3
- Μέσος χρόνος απόκρισης = (9+7 + 1+4 + 0+1 + 2+4)/4 = 7

2.4 *Ασκήσεις με εικονικές και φυσικές διευθύνσεις*

2.4.1 Άσκηση 1

Για κάθε μία από τις παρακάτω εικονικές διευθύνσεις (οι οποίες είναι σε δεκαδική μορφή), υπολόγισε την εικονική σελίδα και τη μετατόπιση (offset) για μέγεθος σελίδας 4K και 8K: 20000, 32768, 60000.

Απάντηση

Για αναπαράσταση σελίδων των 4K χρειάζονται 12bit (2^{12}=4096). Οι σελίδες που δίνονται δεν ξεπερνούν το νούμερο 65535 το οποίο ισούτε με 2^{16}. Συμπεραίνουμε, λοιπόν, ότι οι διευθύνσεις έχουν 16bit εκ' των οποίων τα 4 χρησιμοποιούνται για τον αριθμό σελίδας και τα 12 για την μετατόπιση (offset). Με 4bit για τις σελίδες έχουμε συνολικά 2^4=16 σελίδες.

Η εικονική διεύθυνση 20000 δίνει:
αριθμός σελίδας = 20000 div 4096 = 4 (div το ακέραιο μέρος της διαίρεσης)
μετατόπιση = 20000 - (αριθμός σελίδας)*4096 = 20000 - 4*4096 = 20000 - 16384 = 3616

Η εικονική διεύθυνση 32768 δίνει:
αριθμός σελίδας = 32768 div 4096 = 8
μετατόπιση = 32768 - (αριθμός σελίδας)*4096 = 32768 - 8*4096 = 32768 - 32768 = 0

Η εικονική διεύθυνση 60000 δίνει:
αριθμός σελίδας = 60000 div 4096 = 14
μετατόπιση = 60000 - (αριθμός σελίδας)*4096 = 60000 - 14*4096 = 60000 - 57344 = 2656

Για σελίδες των 8K χρειάζονται 13bit (2^{13}=8192). Έτσι, για τις σελίδες έχουμε 3bit (16 - 13) και συνολικά 2^3=8 σελίδες.

Η εικονική διεύθυνση 20000 δίνει:
αριθμός σελίδας = 20000 div 8192 = 2
μετατόπιση = 20000 - (αριθμός σελίδας)*8192 = 20000 - 2*8192 = 20000 - 16384 = 3616

Η εικονική διεύθυνση 32768 δίνει:
αριθμός σελίδας = 32768 div 8192 = 4
μετατόπιση = 32768 - (αριθμός σελίδας)*8192 = 32768 - 4*8192 = 32768 - 32768 = 0

Η εικονική διεύθυνση 60000 δίνει:
αριθμός σελίδας = 60000 div 8192 = 7
μετατόπιση = 60000 - (αριθμός σελίδας)*8192 = 60000 - 7*8192 = 60000 - 57344 = 2656

Γενικά

Εικονική Διεύθυνση	Μέγεθος σελίδας
offset	Αριθμός Εικονικής σελίδας

Σελίδα	Διευθύνσεις
4	20.479 16.384
3	16.383 12.288
2	12.287 8.192
1	8191 4096
0	4095 0

2.4.2 Άσκηση 2

Ένα σύστημα σελιδοποίησης χρησιμοποιεί διευθύνσεις των 16 bits, με μέγεθος σελίδας 4K και συνολικά 8 πλαίσια φυσικής μνήμης. Δίνονται οι πίνακες σελίδων των διεργασιών P1 και P2.

	P1
0	0
1	4
2	5
3	2

	P2
0	3
1	1
2	7
3	6

Να βρεθούν οι φυσικές διευθύνσεις των παρακάτω λογικών διευθύνσεων:

- Λογική διεύθυνση 15.000 της διεργασίας P1
- Λογική διεύθυνση 12.000 της διεργασίας P2

Απάντηση

ΔΙΕΡΓΑΣΙΑ	ΔΙΕΥΘΥΝΣΗ	ΑΡΙΘΜΟΣ ΣΕΛΙΔΑΣ	ΜΕΤΑΤΟΠΙΣΗ	ΦΥΣΙΚΗ ΔΙΕΥΘΥΝΣΗ
P1	15000	3	2712	10904
P2	12000	2	3808	32480

ΑΝΑΛΥΤΙΚΑ (4K = 4096 bytes)

ΔΙΕΡΓΑΣΙΑ P1

Διεύθυνση 15000:15000/4K=15000/4096 = 3 ολόκληρες σελίδες (οι 0, 1, 2) και στη σελίδα 3 με υπόλοιπο 15000 – 3*4.096= 2.712 από τον πίνακα διεργασίας η σελίδα 3 αντιστοιχεί στο πλαίσιο 2 δηλ. **φυσική διεύθυνση = 2*4.096+2.712= 10.904**

ΔΙΕΡΓΑΣΙΑ P2

Διεύθυνση 12000:12000/4K=12000/4096 = 2 ολόκληρες σελίδες (οι 0 και 1) και στη σελίδα 2 με υπόλοιπο 12.000 – 2*4.096= 3.808 από τον πίνακα διεργασίας η σελίδα 2 αντιστοιχεί στο πλαίσιο 7 δηλ. **φυσική διεύθυνση = 7*4.096+3.808= 32.480**

2.4.3 Άσκηση 3

Μια μηχανή έχει εικονικές διευθύνσεις των 48 bit και φυσικές διευθύνσεις των 32 bit. Οι σελίδες έχουν μέγεθος 8KB. Πόσες καταχωρήσεις απαιτούνται για τον πίνακα σελίδων;

Απάντηση

Το μέγεθος του εικονικού χώρου διευθύνσεων είναι 2^48 bytes. Το μέγεθος κάθε σελίδας είναι 8KB=8192 bytes=2^13 bytes. Άρα το πλήθος σελίδων του πίνακα σελίδων είναι: 2^48/2^13=2^35 σελίδες.

Παρατήρηση: Αν ζητούσε το πλήθος φυσικών πλαισίων της μνήμης αυτό θα ήταν: 2^32/2^13=2^19 φυσικά πλαίσια.

2.5 Ασκήσεις με πολυδιάστατο πίνακα σελίδων

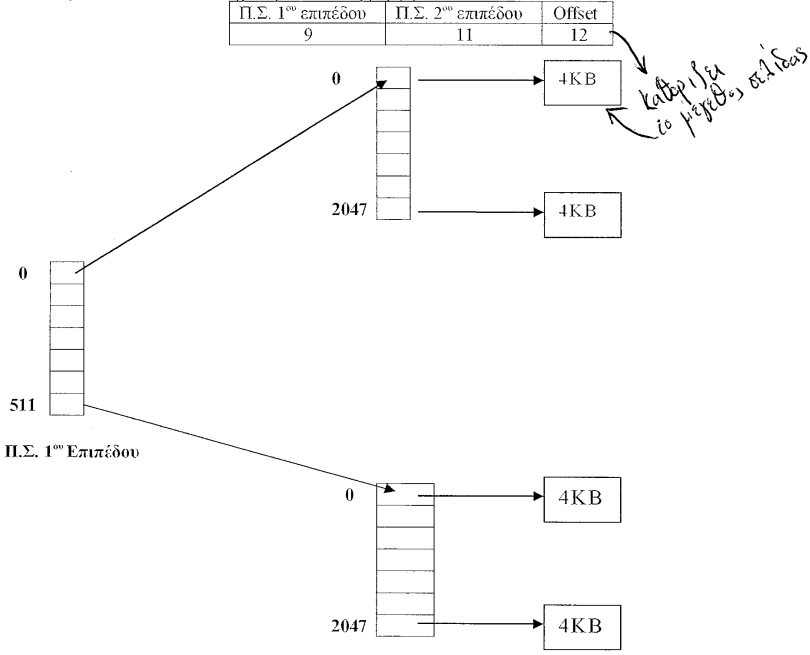
2.5.1 Άσκηση 1

Ένας υπολογιστής με διευθύνσεις των 32 bit χρησιμοποιεί ένα πίνακα σελίδων 2 επιπέδων. Οι εικονικές διευθύνσεις χωρίζονται σε ένα πεδίο 9 bit για τον πίνακα σελίδων 1ου επιπέδου, ένα πεδίο 11 bit για τον πίνακα σελίδων 2ου επιπέδου και μια σχετική διεύθυνση (offset). Πόσο μεγάλες είναι οι σελίδες και πόσες υπάρχουν στο χώρο διευθύνσεων;

Απάντηση

Η κάθε διεύθυνση του Π.Σ. 2 επιπέδων έχει την ακόλουθη μορφή:

Π.Σ. 1ου επιπέδου	Π.Σ. 2ου επιπέδου	Offset
9	11	12



Το μέγεθος σελίδας είναι 4KB και έχουμε 2^20 σελίδες.

Παρατήρηση: Αν έχουμε μόνο 1 πίνακα σελίδων αυτός θα έχει 2^32/2^12=2^20 σελίδες και θα πρέπει να φορτώνει όλο τον πίνακα σελίδων του στην κύρια μνήμη. Αυτό πρακτικά σημαίνει ότι ο πίνακας σελίδων έχει παρά πολύ μεγάλο μέγεθος και συνήθως δεν χωρά στην κύρια μνήμη. Αν χρησιμοποιήσουμε πίνακα σελίδων 2 επιπέδων όπως στο παράδειγμα μας, τότε θα υπάρχουν συνολικά 1 πίνακας σελίδων 1ου επιπέδου και 512 πίνακες σελίδων 2ου επιπέδου, όμως η διεργασία για να εκτελεστεί θα χρειαστεί αρχικά να φορτώσει μόνο 4 πίνακες σελίδων (τον αρχικό πίνακα σελίδων και 3 πίνακες σελίδων 2ου επιπέδου για το text, το data και το stack αντίστοιχα) και αυτό σημαίνει σημαντική μείωση του μεγέθους του πίνακα σελίδων. Αν χρειαστεί και άλλους πίνακες σελίδων στην πορεία σίγουρα ο αριθμός σελίδων θα είναι μικρότερος από τις 2^20 σελίδες.

2.5.2 Άσκηση 2

Υποθέστε ότι μια μηχανή έχει εικονικές διευθύνσεις των 38 bit και φυσικές διευθύνσεις των 32 bit.

α)Ποιο είναι το βασικό πλεονέκτημα ενός πολυεπίπεδου πίνακα σελίδων;

β)Με ένα πίνακα σελίδων 2 επιπέδων, σελίδες των 16 KB και καταχωρίσεις των 4 bytes πόσα bit πρέπει να εκχωρηθούν στο πεδίο του κορυφαίου επιπέδου και πόσα στο πεδίο του επόμενου επιπέδου;

Απάντηση

Ο πολυεπίπεδος πίνακας σελίδων μειώνει σημαντικά τον αριθμό των σελίδων που πρέπει να φορτωθούν στην κύρια μνήμη γιατί αρχικά έχουμε μόνο 4 πίνακες σελίδων, ενώ ο πίνακας σελίδων ενός επιπέδου έχει πολύ μεγαλύτερο μέγεθος και δεν χωρά στην κύρια μνήμη.

Π.Σ. 1ου επιπέδου	Π.Σ. 2ου επιπέδου	offset

Το μέγεθος σελίδας είναι 16KB=16384 bytes =2^14 bytes. Άρα το offset που καθορίζει το μέγεθος σελίδας είναι 14 bit
Άρα 38-14=24 bit μένουν για τα 2 πρώτα επίπεδα.

Μέσα σε κάθε σελίδα ο αριθμό φυσικού πλαισίου στο οποίο αυτή απεικονίζεται έχει μέγεθος 4 bytes άρα 2^14/2^2=2^12 φυσικά πλαίσια δηλ. κάθε πίνακας σελίδων 2ου επιπέδου πρέπει να δείχνει σε 2^12 φυσικά πλαίσια. συνεπώς χρειαζόμαστε 12 bit για τον Π.Σ. 2ου επιπέδου και έτσι απομένουν 12 bit για τον Π.Σ. 1ου επιπέδου

2.5.3 Άσκηση 3

Ένα σύστημα εικονικής μνήμης χρησιμοποιεί σελιδοποίηση με εικονικές διευθύνσεις μεγέθους 64 bits και μέγεθος σελίδας 16KB. Κάθε είσοδος στον πίνακα σελίδων απαιτεί 128 bits. Ο πίνακας σελίδων πρέπει να χωρά σε μία σελίδα. Πόσα επίπεδα πινάκων σελίδων απαιτούνται και πόσες είσοδοι υπάρχουν στον πίνακα σελίδων σε κάθε επίπεδο;

Απάντηση

Για εικονικές διευθύνσεις μεγέθους 64 bits και μέγεθος σελίδας 16KB η κατανομή είναι 14 bits για τη μετατόπιση εντός της σελίδας και απομένουν 50 bits (=64-14) για τα επίπεδα των πινάκων σελίδων.
Για εισόδους των 128 bits = 16 bytes και μέγεθος σελίδας 16KB προκύπτει πλήθος 1K=1024 εισόδων σε κάθε πίνακα σελίδων που χωρά σε μια σελίδα.
Αυτό σημαίνει ότι κάθε επίπεδο μπορεί να χρησιμοποιήσει κατά μέγιστο 10 bits και το σύστημα θα απαιτήσει 5 επίπεδα πινάκων σελίδων με 1024 εισόδους σε κάθε επίπεδο.

2.5.4 Άσκηση 4

Ένα σύστημα εικονικής μνήμης χρησιμοποιεί χώρο λογικών διευθύνσεων που αποτελείται από 256 σελίδες μεγέθους 4096 bytes η κάθε μία. Το μέγιστο μέγεθος της φυσικής μνήμης είναι 16 πλαίσια.

1. Πόσα bits χρησιμοποιούνται για το σχηματισμό μιας λογικής διεύθυνσης;
2. Πόσα bits χρησιμοποιούνται για το σχηματισμό μιας φυσικής διεύθυνσης;
3. Αν κάθε είσοδος του πίνακα σελίδων μιας διεργασίας απαιτεί 4 bytes πόσες κατά μέγιστο είσοδοι μπορούν να υπάρξουν, με την προϋπόθεση ότι ο πίνακας σελίδων καταλαμβάνει μία σελίδα;

Απάντηση

1. 2^8*2^12=2^20 → 20 bits
2. 16 πλαίσια =16*4096 bytes =2^4*2^12=2^16 → 16 bits
3. μέγεθος πίνακα σελίδων =4096 bytes / 4 bytes =1024 είσοδοι

2.5.5 Άσκηση 5

Μια μηχανή έχει εικονικές διευθύνσεις των 32 bit. Οι σελίδες έχουν μέγεθος 4KB. Το πρόγραμμα (text) και τα δεδομένα (data) χωράνε μαζί στη χαμηλότερη σελίδα (0-4095). Η στοίβα (stack) χωρά στην υψηλότερη σελίδα. Πόσες σελίδες θα έχει ο πίνακας σελίδων αν χρησιμοποιηθεί πίνακας σελίδων ενός επιπέδου; Πόσες σελίδες θα έχει ο πίνακας σελίδων αν χρησιμοποιηθεί πίνακας σελίδων δύο επιπέδων με 10 bit σε κάθε τμήμα.

Απάντηση

Το μέγεθος του εικονικού χώρου διευθύνσεων είναι 2^32 bytes. Το μέγεθος κάθε σελίδας είναι 4KB=4096 bytes=2^12 bytes. Άρα το πλήθος σελίδων του πίνακα σελίδων ενός επιπέδου είναι: 2^32/2^12=2^20 σελίδες. Αντίστοιχα στον πίνακα σελίδων δύο επιπέδων ο πίνακας 1ου επιπέδου θα έχει 2^10 σελίδες. Η κάθε μια από αυτές θα δείχνει σε ένα πίνακα σελίδων 2ου επιπέδου (συνολικά θα έχουμε 2^10 πίνακες 2ου επιπέδου). Ο κάθε πίνακας σελίδων 2ου επιπέδου θα έχει 2^10 σελίδες. Θα χρησιμοποιούν αρχικά μόνο 3 πίνακες σελίδων

2.6 Ασκήσεις με TLB

2.6.1 Άσκηση 1

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;

Απάντηση

- 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) στο TLB : 50ns + 400ns + 400ns = 850ns
- Πραγματικός χρόνος προσπέλασης με απλή σελιδοποίηση και TLB :
  - 450*80% + 850*20% = 530ns
- Speed-up = 800/530 = 1.51

2.6.2 Άσκηση 2

Ένας Η/Υ του οποίου οι διεργασίες έχουν 1024 σελίδες στο χώρο διευθύνσεων φυλά τον πίνακα σελίδων του στην κύρια μνήμη. Το overhead που απαιτείται για το διάβασμα μιας λέξης από τον πίνακα σελίδων είναι 5 nsec. Για τη μείωση του overhead ο Η/Υ χρησιμοποιεί μια TLB με 32 ζεύγη (εικονική σελίδα, φυσικό πλαίσιο) και εκτελεί μια αναζήτηση σε 1 nsec. Ποιο hit rate απαιτείται για να μειώσει το μέσο overhead στα 2 nsec;

Απάντηση

Ο τύπος που δίνει το μέσο χρόνο επίδοσης στη συσχετιστική μνήμη (TBL) δίνεται από τον τύπο:

$$t_\mu = \frac{h}{100}\cdot t_c + \left(1-\frac{h}{100}\right)\cdot(t_c + t_p)$$

t_μ: Χρόνος μέσης επίδοσης
t_p: Χρόνος προσπέλασης στον πίνακα σελίδων
t_c: Χρόνος προσπέλασης στη συσχετιστική μνήμη (TBL)
h: Ποσοστό επιτυχίας

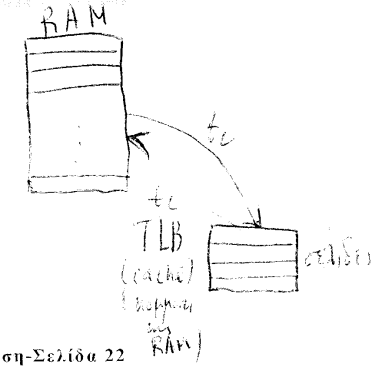
Εδώ έχουμε: t_p=5nsec, t_c=1nsec, t_μ=2nsec Κάνουμε αντικατάσταση:

$$2 = \frac{h}{100}\cdot 1 + \left(1-\frac{h}{100}\right)\cdot 6 \Rightarrow 2 = 0.01h + 6 - 0.06h = 0.05h = 4 \Rightarrow h = 80$$

Άρα το ποσοστό επιτυχίας πρέπει να είναι 80%.

Στον TLB αποθηκεύονται οι σελίδες που χρησιμοποιούνται πιο συχνά

Η διεργασία όταν χρειάζεται μια σελίδα ψάχνει πρώτα στον TLB και είαν δεν έχει τον φορτώνει στον RAM



2.6.3 Άσκηση 3

8.9 Θεωρήστε ένα σύστημα σελιδοποίησης με τον πίνακα σελίδων αποθηκευμένο στη μνήμη.
α. Αν μία αναφορά στη μνήμη απαιτεί 200 ns, πόσο χρόνο απαιτεί μία αναφορά σε μνήμη με χρήση σελίδας;
β. Αν προσθέσουμε TLBs, και 75 τοις εκατό όλων των αναφορών στους πίνακες σελίδων βρίσκονται στις TLBs, ποιος είναι ο πραγματικός χρόνος αναφοράς σε μνήμη; (Υποθέστε ότι η αναζήτηση μίας εγγραφής στον πίνακα σελίδων της TLB διαρκεί μηδενικό χρόνο, αν η εγγραφή υπάρχει).

Απάντηση
α. Αναφορά = αναφορά σε πίνακα σελίδων + αναφορά στο πλαίσιο μνήμης = 200 + 200 = 400ns

β. Έστω p η πιθανότητα επιτυχίας στην TLB. Τότε
χρόνος = p*(χρόνος TLB + χρόνος στο πλαίσιο μνήμης) + (1-p)*(χρόνος TLB + χρόνος στον πίνακα σελίδων + χρόνος στο πλαίσιο μνήμης)= 0,75*(0+200)+0,25(0+200+200)= 150+100=250 ns.

2.7 Ασκήσεις με Παρακολούθηση Κενών Block Δίσκου

2.7.1 Άσκηση 1

Η παρακολούθηση του ελεύθερου χώρου στο δίσκο μπορεί να γίνει με τη χρήση μιας λίστας ελεύθερων μπλοκ ή με ένα χάρτη bit (bitmap). Οι διευθύνσεις του δίσκου απαιτούν Δ bit. Αν ένας δίσκος έχει χωρητικότητα Μ μπλοκ από τα οποία Ε είναι ελεύθερα, δώστε τη συνθήκη με την οποία η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο από χάρτη bit. Αν το Δ έχει τιμή 16bit εκφράστε την απάντηση σας ως ποσοστό του χώρου δίσκου ο οποίος πρέπει να είναι ελεύθερος.

Απάντηση

Στο χάρτη ψηφίων (bitmap) χρησιμοποιείται 1 bit για κάθε μπλοκ το οποίο υποδηλώνει την κατάσταση του μπλοκ (συγκεκριμένα έχει την τιμή 1 αν το μπλοκ είναι κενό ή 0 αν το μπλοκ είναι γεμάτο). Επειδή ο δίσκος έχει Μ μπλοκ άρα το bitmap έχει μέγεθος Μ bits. Η λίστα ελεύθερων μπλοκ έχει μέγεθος Δ•Ε bits γιατί παρακολουθούμε μόνο τα κενά μπλοκ τα οποία σε πλήθος είναι Ε (σε κάθε ειδικό μπλοκ θεωρούμε ότι βρίσκεται μόνο ένας αριθμός κενού μπλοκ). Η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο από το χάρτη bit εφόσον Δ•Ε<Μ. Εναλλακτικά η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο σε σχέση με το χάρτη ψηφίων εφόσον Ε/Μ<1/Δ όπου το Ε/Μ είναι το ποσοστό κενών μπλοκ. Αν Δ=16 bit τότε Ε/Μ<1/Δ=0,0625 ή αν 6.25% του χώρου του σκληρού δίσκου είναι κενός.

Παρατήρηση: Το αποτέλεσμα στο οποίο καταλήγουμε επιβεβαιώνει και τη θεωρία η οποία λέει ότι η λίστα ελεύθερων μπλοκ προτιμάται από το χάρτη ψηφίων όταν ο δίσκος έχει πολύ λίγα κενά μπλοκ.

2.7.2 Άσκηση 2

Ένας ελεγκτής του συστήματος αρχείων έχει δημιουργήσει τους μετρητές του όπως φαίνονται παρακάτω.

Αριθμός Μπλοκ:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Σε Χρήση:	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0
Ελεύθεροι:	0	0	0	1	1	1	0	0	0	1	0	1	1	0	1

Υπάρχουν λάθη; Αν ναι, είναι σοβαρά; Γιατί;

Απάντηση

Υπάρχουν λάθη στα μπλοκ 1, 5 και 6. Τα μπλοκ 1 και 6 δεν εμφανίζονται ούτε στη λίστα των μπλοκ σε χρήση ούτε στη λίστα των ελεύθερων. Αυτό το πρόβλημα δεν είναι σοβαρό, διότι μπορεί απλά να λυθεί τοποθετώντας τα μπλοκ στη λίστα των ελεύθερων, δηλαδή, κάνοντας το αντίστοιχο bit 1.

Το πρόβλημα με το μπλοκ 5 είναι πιο σοβαρό διότι το μπλοκ εμφανίζεται και στις δύο λίστες.

2.8 Ασκήσεις με Παρακολούθηση Κενών Διαμερισμάτων Μνήμης

2.8.1 Άσκηση 1

Έχουμε μνήμη μεγέθους 128 MB η οποία εκχωρείται σε μονάδες των n bytes. Για τη συνδεδεμένη λίστα υποθέστε ότι η μνήμη αποτελείται από μια εναλλασσόμενη ακολουθία τμημάτων και οπών με μέγεθος λίστας 64KB (μέγεθος όλης της λίστας). Υποθέστε ακόμα ότι κάθε κόμβος στη συνδεδεμένη λίστα χρειάζεται μια διεύθυνση μνήμης των 32 bit, ένα πεδίο των 16 bit για το μήκος και ένα πεδίο των 16 bit για τον επόμενο κόμβο. Πόσα byte **αποθηκευτικού χώρου χρειάζονται σε κάθε περίπτωση; Ποια μέθοδος είναι καλύτερη** (bitmap ή λίστα);

Απάντηση

Υπάρχουν 2 τρόποι παρακολούθησης των διαμερισμάτων (κύριας) μνήμης. **Ο 1ος τρόπος είναι ο χάρτης ψηφίων (bitmap) ο οποίος χρησιμοποιεί ένα bit για κάθε μονάδα αποθήκευσης (διαμέρισμα μνήμης). Ο 2ος τρόπος είναι η συνδεδεμένη λίστα η οποία χρησιμοποιεί ένα κόμβο για κάθε μονάδα αποθήκευσης (διαμέρισμα μνήμης).**

- Η συνολική μνήμη είναι 128 MB=128 MB x1024 KB x 1024 bytes=2^27 bytes
- Η μνήμη αυτή χωρίζεται σε διαμερίσματα (allocation units) των n bytes, άρα υπάρχουν συνολικά 2^27/n διαμερίσματα μνήμης

✓ Αν χρησιμοποιήσουμε τη μέθοδο bitmap χρειαζόμαστε 1 bit για το διαμέρισμα μνήμης, άρα χρειαζόμαστε συνολικά 2^27/n bit ή αλλιώς χρειαζόμαστε 2^24/n bytes για το χάρτη ψηφίων (Μέγεθος του χάρτη ψηφίων)

✓ Αν χρησιμοποιήσουμε τη μέθοδο της συνδεδεμένης λίστας το μέγεθος όλης της λίστας είναι 64KB = 64 KB x 1024 bytes= =2^16 bytes. Διαιρώντας το μέγεθος όλης της μνήμης με το μέγεθος κάθε κόμβου υπολογίσουμε ότι υπάρχουν συνολικά 2^27/2^16= 2^11 κόμβοι στη συνδεδεμένη λίστα. Το μέγεθος κάθε κόμβου είναι 64 bit=32+16+16 ή 8 byte, άρα το μέγεθος όλης της λίστας σε bytes είναι: 2^11 κόμβοι x 8 byte/κόμβο= 2^14 bytes

➢ Για να είναι προτιμότερη η μέθοδος της συνδεδεμένης λίστας πρέπει 2^14<2^24/n⇒n<2^10=1 KB, αλλιώς αν n>2^10 τότε είναι προτιμότερος ο χάρτης ψηφίων

Παρατήρηση

Η δομή κάθε κόμβου της λίστας είναι η ακόλουθη:

32 bit Διεύθυνση Διαμερίσματος	16 bit Μήκος Διαμερίσματος	16 bit δείκτης επόμενου κόμβου

Δίνω με λίστα παραπλανώμαι μόνο για μένα
ενώ στη μνήμη με λίστα παραπλανώμαι όλα.

2.9 *Διάφορες Ασκήσεις με Σελιδοποίηση*

2.9.1 Άσκηση 1

Υποθέστε ότι έχουμε μία μνήμη με σελιδοποίηση κατ' απαίτηση. Ο πίνακας σελίδων διατηρείται σε καταχωρητές. Η εξυπηρέτηση ενός σφάλματος σελίδας διαρκεί 8 ms αν είναι διαθέσιμο ένα κενό πλαίσιο ή αν η σελίδα που αντικαθίσταται δεν είναι τροποποιημένη και 20 ms αν είναι τροποποιημένη. Ο χρόνος της προσπέλασης της μνήμης είναι 100 ns.
Υποθέστε ότι η σελίδα που πρόκειται να αντικατασταθεί είναι τροποποιημένη το 70% των φορών. Ποιος είναι ο μέγιστος αποδεκτός ρυθμός σφαλμάτων σελίδας για πραγματικό χρόνο πρόσβασης όχι μεγαλύτερο από 200 ns;

Απάντηση

$200\text{ns} \geq (1\text{-p}) \times 100\text{ns} + (0.3 \times \text{p}) \times 8\text{ms} + (0.7 \times \text{p}) \times 20\text{ms}$
Από την παραπάνω ανίσωση προκύπτει ότι $\text{p} \leq 0.000006$

2.9.2 Άσκηση 2

Θεωρήστε ένα σύστημα σελιδοποίησης κατ' απαίτηση με ένα δίσκο σελιδοποιημένο που έχει μέσο χρόνο προσπέλασης και μεταφοράς 20 ms. Οι διευθύνσεις μεταφράζονται μέσω ενός πίνακα σελίδων στην κύρια μνήμη, με χρόνο προσπέλασης 1 μs ανά προσπέλαση μνήμης. Έτσι, κάθε αναφορά μνήμης μέσω του πίνακα σελίδων απαιτεί δύο προσπελάσεις. Για να βελτιωθεί αυτός ο χρόνος, έχουμε προσθέσει μία συσχετιστική μνήμη που μειώνει το χρόνο προσπέλασης σε μία αναφορά μνήμης αν η εγγραφή στον πίνακα σελίδων βρίσκεται στη συσχετιστική μνήμη.
Υποθέστε ότι 80% των προσπελάσεων βρίσκονται στη συσχετιστική μνήμη και ότι, από τις υπόλοιπες, 10% (ή 2% του συνόλου) προκαλεί σφάλματα σελίδας. Ποιος είναι ο πραγματικός χρόνος πρόσβασης;

Απάντηση

Χρόνος πρόσβασης = $0.8 \times 1\mu\text{s} + 0.18 \times (1\mu\text{s} + 1\mu\text{s}) + 0.02 \times (1\mu\text{s} + 20000\mu\text{s} + 1\mu\text{s}) =$
$0.8 + 0.36 + 400.04 = 401.20\mu\text{s}$

2.9.3 Άσκηση 3

Ένας υπολογιστής παρέχει σε κάθε διεργασία χώρο διευθύνσεων 65536 bytes που διαιρείται σε σελίδων των 4096 bytes. Ένα συγκεκριμένο πρόγραμμα έχει ένα κείμενο μεγέθους 32768 bytes, δεδομένα μεγέθους 16386 bytes και μια στοίβα μεγέθους 15870 bytes. Θα μπορέσει αυτό το πρόγραμμα να χωρέσει στην μνήμη; Αν το μεγέθος της σελίδος ήταν 512 bytes, θα ταιριάζει; Θυμηθείτε ότι μια σελίδα δεν μπορεί να περιέχει κομμάτια δύο διαφορετικών τμημάτων.

Απάντηση

Για σελίδες των 4096 bytes, χρειάζονται:
- Κείμενο: 32768 / 4096 = 8 σελίδες
- Δεδομένα: 16386 / 4096 = 4,000488281 = 5 σελίδες (για το λίγο ακόμη που χρειάζεται παίρνει ολόκληρη την επιπλέον σελίδα)
- Στοίβα: 15870 / 4096 = 3,874511719 = 4 σελίδες

Συνολικά χρειάζεται 17 σελίδες. Στη μνήμη είναι διαθέσιμες 65536 / 4096 = 16 σελίδες. Άρα το πρόγραμμα δεν χωράει.

Για σελίδες των 512 bytes, χρειάζονται:
- Κείμενο: 32768 / 512 = 64 σελίδες
- Δεδομένα: 16386 / 512 = 32,00390625 = 33 σελίδες
- Στοίβα: 15870 / 512 = 30,99609375 = 31 σελίδες

Συνολικά χρειάζεται 128 σελίδες. Στη μνήμη είναι διαθέσιμες 65536 / 512 = 128 σελίδες. Άρα το πρόγραμμα χωράει.

2.9.4 Άσκηση 4

Ένα σύστημα χρησιμοποιεί ως εικονικές διευθύνσεις 2048 σελίδες μεγέθους 256 bytes η κάθε μία και αντιστοιχείται σε μια φυσική μνήμη 512 πλαισίων. Η μικρότερη μονάδα προσπέλασης είναι 1 byte.
1. Μια διεργασία P_1 χρησιμοποιεί 1.053 bytes. Πόσα πλαίσια σελίδων θα απαιτήσει από τη MMU (Memory Management Unit – μονάδα διαχείρισης μνήμης);
2. Μια άλλη διεργασία P_2 απαιτεί 4.000 bytes. Θεωρείστε ότι η P_2 έχει αποκτήσει ήδη μέγεθος φυσικής μνήμης 2.048 bytes, που ξεκινούν από τη διεύθυνση 0. Έστω ότι ο επεξεργαστής χρειάζεται να προσπελάσει τη διεύθυνση: 10^0 σελίδα, μετατόπιση 34 στο πρόγραμμα της διεργασίας P_2. Μπορεί να υπάρξει σφάλμα σελίδας; Δικαιολογείστε την απάντησή σας.

Απάντηση

1.

$\lceil 1053/256 \rceil = 5$ πλαίσια σελίδων

2.

Α τρόπος:
Η ζητούμενη εικονική διεύθυνση βρίσκεται στη σελίδα 10 και σε μετατόπιση 34 από την αρχή της άρα είναι η εικονική διεύθυνση 10•256+34 =2.594
Όλη η διεργασία έχει μέγεθος 4.000 bytes και καταλαμβάνει 4.000:256=15,62=16 σελίδες οι οποίες χωράνε όλες στα 512 φυσικά πλαίσια. Όμως έχουν ήδη φορτωθεί σε φυσικά πλαίσια 2.048 bytes δηλ. 2.048:256=7,98=8 σελίδες. Άρα η διεύθυνση 2594 δεν βρίσκεται ανάμεσα σε αυτές που έχουν φορτωθεί σε φυσικά πλαίσια μνήμης άρα θα προκληθεί σφάλμα σελίδας.

Β τρόπος
Η ζητούμενη εικονική διεύθυνση βρίσκεται στη σελίδα 10. Μέχρι τώρα έχουν φορτωθεί 2.048:256=7,98=8 σελίδες άρα δεν βρίσκεται σε σελίδα που έχει ήδη φορτωθεί στη μνήμη και συνεπώς θα προκληθεί σφάλμα σελίδας

2.9.5 Άσκηση 5

Ένα σύστημα χρησιμοποιεί διευθύνσεις των 32 bits και έχει κεντρική μνήμη 4MB. Το μέγεθος σελίδας είναι 1K. Ποιο είναι το μέγεθος του πίνακα σελίδων;

Απάντηση

Ο πίνακας σελίδων περιέχει το πεδίο του αριθμού σελίδας της εικονικής διεύθυνσης. Έτσι το πλήθος των γραμμών στον πίνακα σελίδων είναι ίσο με τον αριθμό των εικονικών σελίδων:

- Εικονικές σελίδες = $2^{32}/2^{10}=2^{22}$=4M σελίδες

Άρα το μέγεθος του πίνακα σελίδων είναι: 2^{22}*22 bits =11,5 Mbytes

2.9.6 Άσκηση 6

Ένα σύστημα χρησιμοποιεί χώρο διευθύνσεων 32 bits και έχει μέγεθος σελίδας 8K. Ο πίνακας σελίδων βρίσκεται εξ ολοκλήρου στο υλικό και κάθε είσοδός του έχει μήκος 32 bits. Όταν μια διεργασία καθίσταται εκτελέσιμη ο πίνακας σελίδων αντιγράφεται από το υλικό στη μνήμη με ταχύτητα 100nsec για κάθε είσοδό του. Αν κάθε διεργασία εκτελείται για 100msec (περιλαμβάνεται ο χρόνος φόρτωσης του πίνακα σελίδων), να βρεθεί το ποσοστό % του χρόνου της CPU που αφιερώνεται για τη φόρτωση του πίνακα σελίδων, για κάθε διεργασία.

Απάντηση

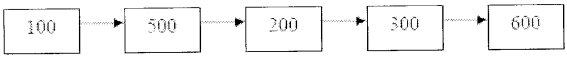
- Μέγεθος σελίδας 8KB=2^13 και απομένουν 32-13=19 bits για τον αριθμό σελίδας
- Ο πίνακας σελίδων έχει 2^19 εισόδους
- Χρόνος φόρτωσης του πίνακα σελίδων = 2^19 * 100nsec = 52,4288msec
- Ποσοστό χρόνου = 52,4288/100 = 52,4288%
- Δηλαδή περίπου το 52% του χρόνου αφιερώνεται για τη φόρτωση του πίνακα σελίδων

2.10 Ασκήσεις με Οπές Μνήμης

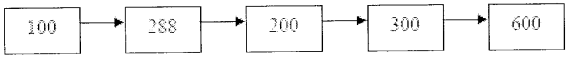
2.10.1 Άσκηση 1

8.3 Δίδονται πέντε διαμερίσματα μνήμης των 100 KB, 500 KB, 200 KB, 300 KB και 600 KB (με αυτή τη σειρά). Πώς θα τοποθετούσαν τέσσερις διεργασίες των 212 KB, 417 KB, 112 KB και 426 KB (με αυτή τη σειρά) οι αλγόριθμοι πρώτης τοποθέτησης, βέλτιστης τοποθέτησης και χειρότερης τοποθέτησης; Ποιος αλγόριθμος κάνει αποδοτικότερη χρήση της μνήμης;

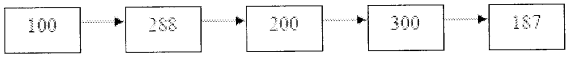
Απάντηση
Η αρχική λίστα των διαθέσιμων διαμερισμάτων μνήμης είναι η εξής:



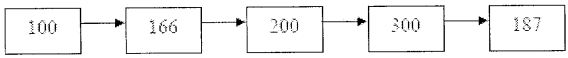
Α. Σύμφωνα με τον αλγόριθμο πρώτης τοποθέτησης η διεργασία των 212 KB θα τοποθετηθεί στο 2ο διαμέρισμα (500KB) από το οποίο θα περισσέψουν 282 KB. Η νέα λίστα έχει ως εξής:



Η διεργασία των 417 KB θα τοποθετηθεί στο 5ο διαμέρισμα (600KB) από το οποίο θα περισσέψουν 183 KB. Η νέα λίστα έχει ως εξής:

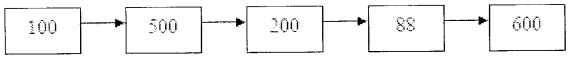


Η διεργασία των 112 KB θα τοποθετηθεί στο 2ο διαμέρισμα (288 KB) από το οποίο θα περισσέψουν 160 KB. Η νέα λίστα έχει ως εξής:

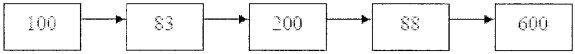


Η διεργασία των 426 KB δεν χωρά σε κάποιο από τα διαθέσιμα διαμερίσματα μνήμης.

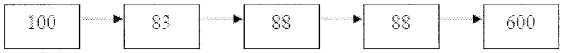
Β. Σύμφωνα με τον αλγόριθμο βέλτιστης τοποθέτησης η διεργασία των 212 KB θα τοποθετηθεί στο 4ο διαμέρισμα (300 KB) από το οποίο θα περισσέψουν 88 KB. Η νέα λίστα έχει ως εξής:



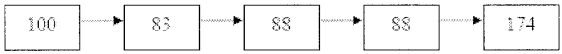
Η διεργασία των 417 KB θα τοποθετηθεί στο 2ο διαμέρισμα (500 KB) από το οποίο θα περισσέψουν 83 KB. Η νέα λίστα έχει ως εξής:



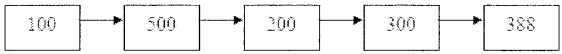
Η διεργασία των 112 KB θα τοποθετηθεί στο 3ο διαμέρισμα (200 KB) από το οποίο θα περισσέψουν 88 KB. Η νέα λίστα έχει ως εξής:



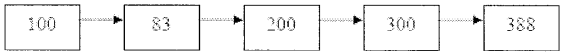
Η διεργασία των 426 KB θα τοποθετηθεί στο 5ο διαμέρισμα (600 KB) από το οποίο θα περισσέψουν 174 KB. Η νέα λίστα έχει ως εξής:



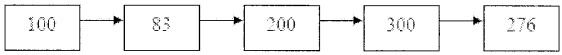
Γ. Σύμφωνα με τον αλγόριθμο χειρότερης τοποθέτησης η διεργασία των 212 KB θα τοποθετηθεί στο 5ο διαμέρισμα (600 KB) από το οποίο θα περισσέψουν 388 KB. Η νέα λίστα έχει ως εξής:



Η διεργασία των 417 KB θα τοποθετηθεί στο 2ο διαμέρισμα (500 KB) από το οποίο θα περισσέψουν 83 KB. Η νέα λίστα έχει ως εξής:



Η διεργασία των 112 KB θα τοποθετηθεί στο 5ο διαμέρισμα (388 KB) από το οποίο θα περισσέψουν 276 KB. Η νέα λίστα έχει ως εξής:



Η διεργασία των 426 KB δεν χωρά σε κάποιο από τα διαθέσιμα διαμερίσματα μνήμης. Άρα αποδοτικότερη χρήση της μνήμης κάνει ο αλγόριθμος βέλτιστης τοποθέτησης.

2.10.2 Άσκηση 2

Θεωρείστε ένα σύστημα εναλλαγής στο οποίο η μνήμη περιέχει τα ακόλουθα μεγέθη κενών κατά σειρά: 10K, 4K, 20K, 18K, 7K, 9K, 12K και 15K. Ποιο κενό χρησιμοποιείται για κάθε επιτυχημένη απαίτηση που έχει μέγεθος,

a. 12K
b. 10K
c. 9K

με τον αλγόριθμο πρώτης τοποθέτησης; Επαναλάβατε την άσκηση για τους αλγόριθμους της καλύτερης, της χειρότερης και της επόμενης τοποθέτησης.

Απάντηση

- Πρώτη Τοποθέτηση
  - Αίτηση 12K

Τοποθετείται στο κενό 20K. Νέα λίστα: 10K, 4K, 8K, 18K, 7K, 9K, 12K, 15K

  - Αίτηση 10K

Τοποθετείται στο κενό 10K. Νέα λίστα: 0K, 4K, 8K, 18K, 7K, 9K, 12K, 15K

  - Αίτηση 9K

Τοποθετείται στο κενό 18K. Νέα λίστα: 0K, 4K, 8K, 9K, 7K, 9K, 12K, 15K

- Καλύτερη Τοποθέτηση
  - Αίτηση 12K

Τοποθετείται στο κενό 12K. Νέα λίστα: 10K, 4K, 20K, 18K, 7K, 9K, 0K, 15K

  - Αίτηση 10K

Τοποθετείται στο κενό 10K. Νέα λίστα: 0K, 4K, 20K, 18K, 7K, 9K, 0K, 15K

  - Αίτηση 9K

Τοποθετείται στο κενό 9K. Νέα λίστα: 0, 4K, 20K, 18K, 7K, 0K, 0K, 15K

- Χειρότερη Τοποθέτηση
  - Αίτηση 12K

Τοποθετείται στο κενό 20K. Νέα λίστα: 10K, 4K, 8K, 18K, 7K, 9K, 12K, 15K

  - Αίτηση 10K

Τοποθετείται στο κενό 18K. Νέα λίστα: 10K, 4K, 8K, 8K, 7K, 9K, 12K, 15K

  - Αίτηση 9K

Τοποθετείται στο κενό 15K. Νέα λίστα: 10K, 4K, 8K, 8K, 7K, 9K, 12K, 6K

- Επόμενη Τοποθέτηση
  - Αίτηση 12K

Τοποθετείται στο κενό 20K. Νέα λίστα: 10K, 4K, 8K, 18K, 7K, 9K, 12K, 15K

  - Αίτηση 10K

Τοποθετείται στο κενό 18K. Νέα λίστα: 10K, 4K, 8K, 8K, 7K, 9K, 12K, 15K

  - Αίτηση 9K

Τοποθετείται στο κενό 9K. Νέα λίστα: 10K, 4K, 8K, 8K, 7K, 0K, 12K, 15K

2.11 Ασκήσεις με Αλγόριθμο Τραπεζίτη και Διαχείριση Πόρων

2.11.1 Άσκηση 1

Υπάρχουν 3 τύποι πόρων με πλήθος: R1 = 9, R2 = 3, R3 = 6. Είναι η παρακάτω κατάσταση ασφαλής;

Total	Available	Max	Allocated
R1 R2 R3	R1 R2 R3	R1 R2 R3	R1 R2 R3
9 3 6	0 1 1	P1 3 2 2	P1 1 0 0
		P2 6 1 3	P2 6 1 2
		P3 3 1 4	P3 2 1 1
		P4 4 2 2	P4 0 0 2

Λύση:

Ψάχνουμε να βρούμε μία γραμμή της οποίας οι επιπλέον απαιτήσεις σε πόρους είναι μικρότερες ή ίσες από το Available. Αν φτιάξουμε τον πίνακα Request από τη σχέση Request [i,j] = Max[i,j] - Allocation [i,j], έχουμε.

Request

	R1	R2	R3
P1	2	2	2
P2	0	0	1
P3	1	0	3
P4	4	2	0

Βλέπουμε ότι η P2 έχει μικρότερες απαιτήσεις από τις διαθέσιμες στο Available. Έτσι, δεσμεύει τους πόρους που χρειάζεται και αφού ολοκληρώσει τους ελευθερώνει. Οι πίνακες τώρα είναι όπως παρακάτω.

Προσέχουμε στον πίνακα Available. Η P2 δεσμεύοντας το ένα στιγμιότυπο του πόρου R3, έχει Allocated: 6 1 3 και ο Available: 0 1 0. Ο Available μετά την ολοκλήρωση της P2 είναι το άθροισμα των παραπάνω.

Available	Max	Allocated	Request
R1 R2 R3	R1 R2 R3	R1 R2 R3	R1 R2 R3
6 2 3	P1 3 2 2	P1 1 0 0	P1 2 2 2
	P2 6 1 3	P2 0 0 0	P2 0 0 0
	P3 3 1 4	P3 2 1 1	P3 1 0 3
	P4 4 2 2	P4 0 0 2	P4 4 2 0

Απ' αυτό το σημείο οποιαδήποτε από τις διεργασίες που παραμένουν μπορεί να ολοκληρώσει αφού υπάρχουν οι διαθέσιμοι πόροι.

2.11.2 Άσκηση 2

Θεωρείστε ένα σύστημα με 4 διεργασίες (A, B, C, D) και 5 επαναχρησιμοποιήσιμους πόρους (R0, R1, R2, R3, R4). Η τρέχουσα ανάθεση πόρων (τρέχουσα κατάσταση) στις διεργασίες δίνεται από τον παρακάτω πίνακα:

Εκχώρηση	**R_0**	**R_1**	**R_2**	**R_3**	**R_4**
A	1	0	2	1	1
B	2	0	1	1	0
C	1	1	0	1	0
D	1	1	1	1	0

Θεωρείστε επίσης ότι οι μέγιστες απαιτήσεις των διεργασιών δίνονται από τον παρακάτω πίνακα:

Μέγιστη απαίτηση	**R_0**	**R_1**	**R_2**	**R_3**	**R_4**
A	1	1	2	1	2
B	2	2	2	1	0
C	2	1	3	1	0
D	1	1	2	2	1

Οι διαθέσιμοι προς εκχώρηση πόροι είναι : **0 0 x 1 1**

Να βρεθεί η μικρότερη τιμή **x** για την οποία το σύστημα θα οδηγηθεί σε ασφαλή κατάσταση.

Απάντηση

Αφαιρώντας τις μέγιστες απαιτήσεις από τις εκχωρήσεις πόρων που έχουν γίνει έχουμε :

Υπόλοιπο απαιτούμενων πόρων	R_0	R_1	R_2	**R3**	**R4**
A	0	1	0	0	1
B	0	2	1	0	0
C	1	0	3	0	0
D	0	0	1	1	1

Το διάνυσμα των διαθέσιμων πόρων είναι : **0 0 x 1 1**

Από τη σύγκριση των υπολοίπων των απαιτήσεων με το διάνυσμα αυτό φαίνεται εύκολα ότι η μόνη διεργασία που μπορεί να δεσμεύσει τους υπόλοιπους διαθέσιμους πόρους είναι η διεργασία D αρκεί το διάνυσμα 0 0 x 1 1 να γίνει ίσο με το διάνυσμα της D δηλαδή 0 0 1 1 1 .

Άρα η μικρότερη τιμή του x είναι καταρχήν 1 και οδηγεί τη διεργασία D στην ολοκλήρωσή της και στη συνέχεια στην απελευθέρωση των πόρων που κατέχει. Ακολούθως μπορούν να ολοκληρωθούν όλες οι εναπομένουσες διεργασίες και το σύστημα να οδηγηθεί σε ασφαλή κατάσταση (πιστή εφαρμογή του αλγορίθμου του τραπεζίτη) και επαληθεύεται ότι η καταρχήν τιμή x=1 οδηγεί το σύστημα σε ασφαλή κατάσταση.

2.11.3 Άσκηση 3

Υποθέστε ένα σύστημα με P διεργασίες και R όμοιους επαναχρησιμοποιήσιμους πόρους. Αν κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 2 μονάδες πόρων, να αποδείξετε ότι δεν μπορεί να υπάρξει αδιέξοδο μόνον όταν ισχύει η συνθήκη P<=R-1.

Απάντηση

Αν σε κάποια χρονική στιγμή οι P σε πλήθος διεργασίες έχουν δεσμεύσει η κάθε μια από μια μονάδα του πόρου R και υπάρχει και ακόμη ελεύθερη διαθέσιμη και μία μονάδα του πόρου τότε μία από τις P διεργασίες θα την δεσμεύσει, θα τη χρησιμοποιήσει, θα ολοκληρώσει την εκτέλεσή της και θα επιστρέψει τις 2 μονάδες πόρου στο σύστημα ώστε να συνεχιστεί και η εκτέλεση των υπολοίπων διεργασιών. Δηλαδή σε αυτήν την οριακή περίπτωση θα πρέπει να ισχύει:

$$P+1=R \rightarrow P=R-1$$

Για προφανείς λόγους οποιαδήποτε τιμή πλήθους διεργασιών P < R-1 δεν μπορεί να δημιουργεί αδιέξοδο μια και θα υπάρχουν διαθέσιμες περισσότερες μονάδες του πόρου. Δηλαδή τελικά οι δύο σχέσεις ενσωματώνονται σε μία: P<=R-1

2.11.4 Άσκηση 4

Σε ένα σύστημα υπάρχουν N σε πλήθος ενεργές διεργασίες που διαμοιράζονται M μονάδες ενός επαναχρησιμοποιήσιμου πόρου R. Κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 3 μονάδες του πόρου R. Να βρείτε ποια σχέση πρέπει να έχουν οι παράμετροι N και M ώστε να μην υπάρχει κίνδυνος αδιεξόδου.

Απάντηση

Αν σε κάποια χρονική στιγμή οι N ενεργές διεργασίες έχουν ήδη δεσμεύσει η κάθε μία από 2 μονάδες του πόρου R τότε θα είναι δεσμευμένες 2N μονάδες του πόρου και πρέπει να υπάρχει τουλάχιστον ακόμη μία μονάδα διαθέσιμη ώστε μία από τις N διεργασίες να την αποκτήσει, να τη χρησιμοποιήσει, να ολοκληρώσει την εκτέλεσή της και στη συνέχεια να επιστρέψει τις 3 μονάδες στο σύστημα ώστε να χρησιμοποιηθούν από άλλες διεργασίες που περιμένουν να απελευθερωθούν πόροι για να μπορέσουν να συνεχίσουν την εκτέλεσή τους. Στην οριακή αυτή περίπτωση λοιπόν θα πρέπει να ισχύει :

$$2N+1=M$$

Οποιαδήποτε τιμή M μεγαλύτερη από το όριο που υπολογίστηκε παραπάνω, θα εξασφαλίζει, κατά μείζονα λόγο την αποφυγή του αδιεξόδου δηλαδή τελικά :

$$M >= 2N+1$$

2.12 Ασκήσεις με Αλγόριθμους Αντικατάστασης Σελίδων

2.12.1 Άσκηση 1

Έστω ότι ένα σύστημα έχει 4 φυσικά πλαίσια σελίδας και εμφανίζεται η ακολουθία σελίδων 0,1,7,2,3,2,7,1,0,3. Υπολογίστε το πλήθος σφαλμάτων σελίδας αν χρησιμοποιηθεί ο αλγόριθμος αντικατάστασης σελίδων α)FIFO και β)LRU

Απάντηση

- FIFO

	0	1	7	2	3	2	7	1	0	3
	0	1	7	2	3	3	3	1	0	0
		0	1	7	2	2	2	2	3	3
			0	1	7	7	7	7	2	2
				0	1	1	1	1	7	7
	P	P	P	P	P			P		

Συνολικά 6 λάθη σελίδας.

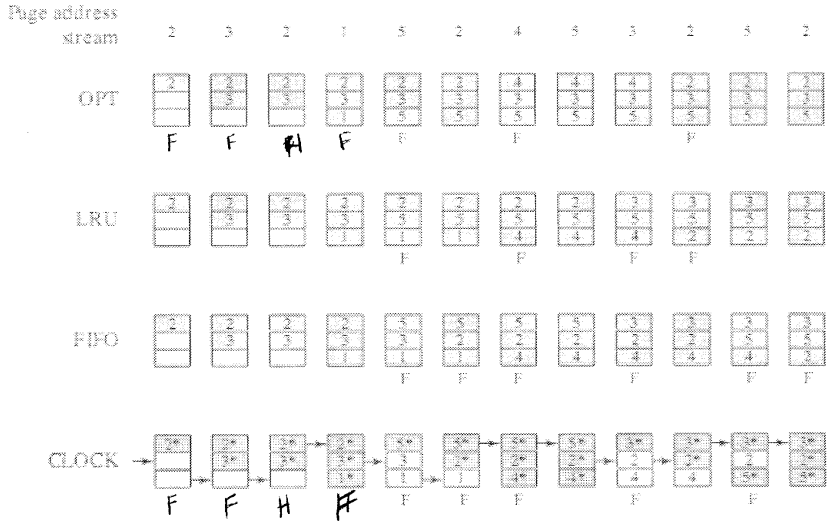
- LRU

	0	1	7	2	3	2	7	1	0	3
	0	1	7	2	3	2	7	1	0	3
		0	1	7	2	3	2	7	1	0
			0	1	7	7	3	2	7	1
				0	1	1	1	3	2	7
	P	P	P	P	P			P	P	

Συνολικά 7 λάθη σελίδας.

2.12.2 Άσκηση 2

- Για την κατανόηση των αλγορίθμων, θα χρησιμοποιήσουμε ένα παράδειγμα
- Μία διεργασία έχει 5 σελίδες (1-5) και της έχει δοθεί στη μνήμη 3 πλαίσια
- Η πρόσβαση στα περιεχόμενα των σελίδων της έχει την εξής σειρά:
  - 2→3→2→1→5→2→4→5→3→2→5→2



2.12.3 Άσκηση 3

Ένας υπολογιστής έχει τέσσερα πλαίσια σελίδας. Ο χρόνος φόρτωσης και τελευταίας προσπέλασης καθώς και οι τιμές των R μαι M bits. Δίνονται παρακάτω.

Σελίδα	Φόρτωση	Προσπέλαση	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

Ποια σελίδα θα αντικατασταθεί σύμφωνα με τους αλγόριθμους:

a. NRU
b. LRU
c. FIFO
d. Δεύτερης ευκαιρίας

Απάντηση

a. NRU: Αντικαθετά σελίδα από την κατηγορία 0 (R=0, M=0), άρα τη σελίδα 2
b. FIFO: Με βάση την φόρτωση θα αφαιρέσει εκείνη με το μικρότερο χρόνο, δηλαδή, τη σελίδα 3
c. LRU: Με βάση την τελευταία προσπέλαση θα αφαιρέσει εκείνη με το μικρότερο χρόνο, δηλαδή, τη σελίδα 1
d. Δεύτερης ευκαιρίας: Με βάση την φόρτωση, υποψήφια είναι η σελίδα 3. Έχει, όμως, R=1, οπότε γίνεται R=0 και βρίσκει την αμέσως επόμενη που είναι η 0. Και αυτή έχει R=1, το οποίο γίνεται 0 και βρίσκει την επόμενη η οποία είναι η 2. Αυτή έχει R=0 οπότε και αφαιρείται.

Ο NRU χωρίζει τις σελίδες σε 4 κατηγορίες

K1	R=0	M=0
K2	R=0	M=1
K3	R=1	M=0
K4	R=1	M=1

και αν συμβεί σφάλμα σελίδας διώχνει σελίδες από πάνω προς τα κάτω (πρώτα από κατηγορία K0, μετά από K1, μετά από K2 κ.λ.π.). Άρα θα διώξει τη σελίδα 2

2.12.4 Άσκηση 4

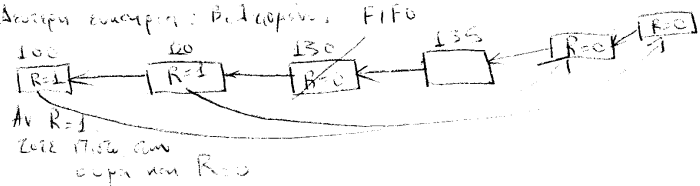
Στον παρακάτω πίνακα σελίδων ποια σελίδα θα αντικατασταθεί με βάση τον αλγόριθμο Working Set αν τ=400 και ο τρέχον εικονικός χρόνος είναι 2204;

Σελίδα	Χρόνος τελευταίας προσπέλασης	R bit
0	2084	1
1	2003	1
2	1980	1
3	1213	0
4	2014	1
5	2020	1
6	2032	1
7	1620	0

Απάντηση

Ο αλγόριθμος αφαιρεί σελίδα η οποία έχει R=0 και ηλικία > τ (ηλικία = Τρέχον εικονικός χρόνος - Χρόνος τελευταίας προσπέλασης).

Η σελίδα με τον μικρότερο χρόνο και R=0 είναι η 3. Η ηλικία της είναι 2204-1213=991>400 άρα αφαιρείται.



2.13 Ασκήσεις με Δίσκους

Στο βιβλίο του Tanenbaum και στις ασκήσεις του δίνει μόνο χρόνο αναζήτησης και χρόνο περιστροφής. Θεωρεί ότι ο χρόνος μετάδοσης ταυτίζεται με το χρόνο περιστροφής. Επίσης, θεωρεί ότι κατά μέσο όρο γίνεται μισή περιστροφή δίσκου μέχρι να βρεθεί το πρώτο μπλοκ του αρχείου. Έτσι, δίνει: χρόνο αναζήτησης, χρόνο περιστροφής, μέγεθος αρχείου και bytes ανά τομέα. Ο τύπος για το χρόνο ανάγνωσης αρχείου γίνεται:

Χρόνος ανάγνωσης αρχείου = χρόνος αναζήτησης + χρόνος περιστροφης/2 + (μέγεθος αρχείου σε bytes / bytes ανα τομεα) x χρόνος περιστροφης

2.13.1 Άσκηση 1

Ένας δίσκος έχει 8 κεφαλές, 256 κυλίνδρους και 128 τομείς ανά κύλινδρο. Αν κάθε τομές έχει 512 bytes.

a. Ποια είναι η χωρητικότητα του δίσκου;
b. Αν ο ρυθμός περιστροφής του είναι 2345 rpm και ο χρόνος αναζήτησης track-σε-track είναι 4msec, πόση πρέπει να είναι η υπερπήδηση;
c. Ποιος είναι ο ρυθμός μεταφοράς δεδομένων;

Απάντηση

α)Χωρητικότητα Δίσκου= Σύνολο Κεφαλών•Αριθμός Κυλίνδρων•Αριθμός Τομέων/Κύλινδρο•bytes/Τομέα=
=8 Κεφαλές •256 Κύλινδροι•128 Τομείς•512 bytes/τομέα=134.217.728 bytes ή 128 Mbytes

β)Μια περιστροφή γίνεται σε χρόνο 1/2345 min ή 60/2345=0,025 sec. Σε μια περιστροφή περνούν κάτω από την κεφαλή 128 τομείς δηλαδή ένας τομέας περνά κάτω από την κεφαλή κάθε 0,025/128=0,000195 sec ή 195 μsec. Στη διάρκεια των 4 msec που είναι το track-to-track περνούν κάτω από την κεφαλή 4 msec/195 μsec 4•10^-3/195•10^-6=20,5 τομείς. (Η υπερπήδηση είναι οι τομείς που διέρχονται κάτω από την κεφαλή όσο χρόνο αυτή μετακινείται από track σε track).

γ)Με ρυθμό 1 τομέα ανά 195 μsec ο δίσκος μπορεί να διαβάσει 1 τομέα/195 μsec=5.128 τομείς/sec. Κάθε τομέας έχει 512 bytes, άρα ο ρυθμός μεταφοράς δεδομένων είναι: 5.128 τομείς/sec•512 bytes/sec=2.625.536 bytes/sec ή 2,5Mbytes/sec

2.13.2 Άσκηση 2

Μια δισκέτα έχει 40 κυλίνδρους. Μια αναζήτηση απαιτεί 6 msec ανά κύλινδρο. Αν δεν γίνει καμία προσπάθεια τοποθέτησης των μπλοκ ενός αρχείων σε γειτονικές θέσεις, δύο μπλοκ τα οποία σχετίζονται λογικά μεταξύ τους (δηλ. είναι διαδοχικά μπλοκ ενός αρχείου) θα βρίσκονται κατά μέσο όρο σε απόσταση 13 κυλίνδρων. Αν όμως το λειτουργικό σύστημα ομαδοποιεί σε clusters σχετιζόμενα μπλοκ, τότε η απόσταση ανάμεσα σε δυο μπλοκ που σχετίζονται λογικά μεταξύ τους μειώνεται σε 2 κυλίνδρους. Πόσος χρόνος απαιτείται για να διαβαστεί ένα αρχείο 100 μπλοκ και στις δύο περιπτώσεις, αν η καθυστέρηση περιστροφής είναι 100 msec και ο χρόνος μεταφοράς είναι 25 msec ανά μπλοκ;

Απάντηση

Ο χρόνος για το κάθε μπλοκ αποτελείται από 3 στοιχεία: Χρόνος Αναζήτησης, Καθυστέρηση Περιστροφής και Χρόνος Μεταφοράς Δεδομένων. Σε όλες τις περιπτώσεις η Καθυστέρηση Περιστροφής και ο Χρόνος Μεταφοράς Δεδομένων είναι ίδιοι και είναι ίσοι με 100+25=125 msec. Αυτό που αλλάζει σε κάθε περίπτωση είναι ο Χρόνος Αναζήτησης.

Αν τα δύο λογικά σχετιζόμενα μπλοκ απέχουν 13 κυλίνδρους τότε ο Χρόνος Αναζήτησης είναι 13•6=78 msec, ενώ τα δύο λογικά σχετιζόμενα μπλοκ απέχουν 2 κυλίνδρους τότε ο Χρόνος Αναζήτησης είναι 2•6=12 msec. Άρα ο συνολικός χρόνος σε κάθε περίπτωση είναι:

- 1η περίπτωση: 78+125=203 msec • 100 =20.300
- 2η περίπτωση: 12+125=137 msec • 100 = 13.700

2.13.3 Άσκηση 3

Θεωρείστε ένα δίσκο με μέσο χρόνο αναζήτησης (average seek time) 8 msec, ρυθμό περιστροφής (rotational rate) 15.000 rpm και 262.144 bytes ανά track. Ποιοι είναι οι ρυθμοί δεδομένων (data rates) για μεγέθη block 1KB, 2KB και 4KB αντίστοιχα .

Απάντηση

Ένας ρυθμός περιστροφής 15.000 rpm σημαίνει ότι ο δίσκος χρειάζεται 60 sec/15.000=0,004 sec=4 msec για μια περιστροφή. Ο μέσος χρόνος προσπέλασης (average access time) σε msec για να διαβαστούν γενικά k bytes είναι:

- Μέσος Χρόνος Προσπέλασης (Διαβάσματος) 1 block= Χρόνος Αναζήτησης (Seek time) + Καθυστέρηση Περιστροφής (Rotation Delay)/2+(Μέγεθος block σε bytes/bytes ανά track)• Καθυστέρηση Περιστροφής (Rotation time)=8 + 4/2 + (k/262.144) •4

Εξετάζουμε τις ακόλουθες περιπτώσεις: Μέσος Χρόνος Προσπέλασης (Διαβάσματος)

- Block size 1KB
  - Read time: 8 + 2 + (1024 / 262144) × 4 = 10,015625 msec
  - Data rate: k (in KB) / read time (in seconds) = 1 / $10,015625 \times 10^{-3}$ = 99.84KB/sec
- Block size 2KB
  - Read time: 8 + 2 + (2048 / 262144) × 4 = 10,03125 msec
  - Data rate: k (in KB) / read time (in seconds) = 2 / $10,03125 \times 10^{-2}$ = 199.38KB/sec
- Block size 4KB
  - Read time: 8 + 2 + (4096 / 262144) × 4 = 10,062088013 msec
  - Data rate: k (in KB) / read time (in seconds) = 4 / $10,062088013 \times 10^{-2}$ = 397.53KB/sec

2.13.4 Άσκηση 4

Ένα συγκεκριμένο σύστημα αρχείων έχει μπλοκ δίσκου με μέγεθος 2KB. Η κεντρική τιμή του μεγέθους αρχείου είναι 1 KB. Αν όλα τα αρχεία ήταν ακριβώς 1 KB, τι ποσοστό χώρου του δίσκου θα πήγαινε χαμένο; Πιστεύετε ότι οι απώλειες σε ένα πραγματικό σύστημα αρχείων είναι μεγαλύτερες ή μικρότερες από το σύστημα αυτό ; Εξηγήστε την απάντηση σας

Απάντηση

Αν όλα τα αρχεία είχαν μέγεθος 1 KB, τότε κάθε block μεγέθους 2 KB θα καταλάμβανε 1 αρχείο και 1 KB χαμένο χώρο. Αυτό οδηγεί σε ποσοστό χαμένου χώρου 50% . Σε ένα πραγματικό σύστημα αρχείων υπάρχουν και μεγάλα και μικρά αρχεία και τα τελευταία χρησιμοποιούν το δίσκο πιο αποδοτικά. Για παράδειγμα ένα αρχείο 32.769 byte καταλαμβάνει 32.769 byte/2048=16,00048 =17 μπλοκ δίνοντας ποσοστό χρήσης του δίσκου 32.768/34.816 (32.768+2.048)=94%

2.14 Ασκήσεις με Αλγόριθμους Χρονοπρογραμματισμού Δίσκου

2.14.1 Άσκηση 1

Να υπολογίσετε το μέσο χρόνο αναζήτησης στο δίσκο (average disk seek time) για τους αλγορίθμους FCFS, SSF, SCAN και C-SCAN όταν έρχονται αιτήσεις για τους κυλίνδρους (tracks): 100, 50, 10, 20 και 75. Ο αρχικός κύλινδρος είναι ο 35 και το bit κατεύθυνσης είναι προς ΕΠΑΝΩ

Απάντηση

α)**FCFS**

35→100→50→10→20→75

Οι αιτήσεις εξυπηρετούνται με βάση τη σειρά άφιξης τους στην ουρά.

Σύνολο κυλίνδρων:65+50+40+10+55=220 και Μέσος Χρόνος Αναζήτησης = 220/5=44

β)**SSF**

Οι αιτήσεις εξυπηρετούνται πηγαίνοντας κάθε φορά στην πλησιέστερο (κοντινότερο) κύλινδρο σε σχέση με αυτόν που βρίσκεται η κεφαλή. Όμως από τον αρχικό κύλινδρο 35 υπάρχουν δύο κύλινδροι που ισαπέχουν: ο 50 και ο 20. Γιαυτό δοκιμάζουμε και τις δύο περιπτώσεις μετακίνησης της κεφαλής και επιλέγουμε αυτή με το μικρότερο συνολικό αριθμό κυλίνδρων.

1η περίπτωση επιλέγουμε τον 50

35→50→75→100→20→10

Σύνολο Κυλίνδρων = 15+25+25+80+10=155 και Μέσος Χρόνος Αναζήτησης = 155/5=31

2η περίπτωση επιλέγουμε τον 20

35→20→10→50→75→100

Σύνολο Κυλίνδρων =15+10+40+25+25=115 και Μέσος Χρόνος Αναζήτησης = 115/5=23

Άρα **επιλέγεται η 2η περίπτωση μετακίνησης που έχει μικρότερο** αριθμό κυλίνδρων.

γ)SCAN

Ο scan αρχίζει από τον κύλινδρο 35 και σαρώνει όλες τις αιτήσεις που βρίσκονται σε μεγαλύτερους κυλίνδρους (προς τα επάνω) πηγαίνοντας κάθε φορά στον πλησιέστερο σε σχέση με αυτόν που είναι η κεφαλή άρα **35→50→75→100**. Στο σημείο αυτό ο βραχίονας έχει φτάσει στο πιο εξωτερικό track. Ο SCAN αλλάζει κατεύθυνση και εξυπηρετεί το ποιο κοντινό track προς τα μέσα (εσωτερικά) άρα: **100→20→10**

Σύνολο κυλίνδρων: 15+25+25+80+10=155 και Μέσος Χρόνος Αναζήτησης = 155/5=31

δ)C-SCAN

Ο C-SCAN δουλεύει αρχικά όπως και ο SCAN δηλαδή αρχίζει από τον κύλινδρο 35 και σαρώνει όλες τις αιτήσεις που βρίσκονται σε μεγαλύτερους κυλίνδρους (προς τα επάνω) πηγαίνοντας κάθε φορά στον πλησιέστερο σε σχέση με αυτόν που είναι η κεφαλή άρα: **35→50→75→100**. Στο σημείο αυτό ο βραχίονας έχει φτάσει στο πιο εξωτερικό track, οπότε τώρα ο βραχίονας τοποθετείται στο αρχικό track (δηλαδή στο track 0) άρα: **100→0** και αρχίζει πάλι να σαρώνει τα tracks προς τα επάνω (έξω) πηγαίνοντας πάντα στο πλησιέστερο συνεπώς: **0→10→20**

Σύνολο κυλίνδρων: 15+25+25+100+10+10=185 και Μέσος Χρόνος Αναζήτησης = 185/5=37

Προσοχή: Παρόλο που έγιναν 6 μετακινήσεις μόνο 5 από αυτές λαμβάνονται υπόψη στον υπολογισμό του μέσου όρου

2.14.2 Άσκηση 2

Αιτήσεις δίσκου φτάνουν για τους κυλίνδρους 10, 22, 20, 2, 40, 6 και 38 με αυτή τη σειρά. Κάθε μετακίνηση σε κύλινδρο διαρκεί 6 msec. Ποιος είναι ο χρόνος αναζήτησης (seek time) που απαιτείται για:

α. Fist Come First Serve

β. SSF

γ. Scan (αλγόριθμος ανελκυστήρα που πηγαίνει αρχικά προς τα πάνω)

δ. C-SCAN (πηγαίνει πάντα προς τα πάνω)

Σε όλες τις περιπτώσεις η κεφαλή βρίσκεται αρχικά στον κύλινδρο 20

Απάντηση

α.First Come First Served

20→10→22→20→2→40→6→38

Το συνολικό άθροισμα κυλίνδρων που γίνεται η μετακίνηση είναι: 10+12+2+18+38+34+32=146 κύλινδροι. Η κάθε μετακίνηση διαρκεί 6 msec άρα ο συνολικός χρόνος είναι 146•6=876 msec

β.SSF

20→20→22→10→6→2→38→40

Το συνολικό άθροισμα κυλίνδρων που γίνεται η μετακίνηση είναι: 0+2+12+4+4+36+32=60 κύλινδροι. Η κάθε μετακίνηση διαρκεί 6 msec άρα ο συνολικός χρόνος είναι 60•6=360 msec

γ.SCAN

20→20→22→38→40→10→6→2

Το συνολικό άθροισμα κυλίνδρων που γίνεται η μετακίνηση είναι: 0+2+16+2+30+4+4=58 κύλινδροι. Η κάθε μετακίνηση διαρκεί 6 msec άρα ο συνολικός χρόνος είναι 58•6=348 msec

2.15 Ασκήσεις με Κόμβο-i

2.15.1 Άσκηση 1

Σε ένα σύστημα UNIX, πόσες αναγνώσεις από το δίσκο πρέπει να γίνουν για να διαβαστεί ο κόμβος-i για το αρχείο /usr/ast/courses/os/handout.t;. Υποθέστε ότι ο κόμβος-i για τον κατάλογο της ρίζας βρίσκεται στη μνήμη. Κανένας από τους άλλους κόμβους-i καταλόγων της διαδρομής δεν βρίσκεται στην μνήμη.

ΑΠΑΝΤΗΣΗ

Έχουμε τις παρακάτω αναγνώσεις:

1. κατάλογος για / (ο κόμβος-i δεν χρειάζεται να διαβαστεί για τη ρίζα, αφού σύμφωνα με την εκφώνηση είναι στη μνήμη ήδη).
2. κόμβος-i για /usr
3. κατάλογος για /usr
4. κόμβος-i για /usr/ast
5. κατάλογος για /usr/ast
6. κόμβος-i για /usr/ast/courses
7. κατάλογος για /usr/ast/courses
8. κόμβος-i για /usr/ast/courses/os
9. κατάλογος για /usr/ast/courses/os
10. κόμβος-i για /usr/ast/courses/os/handout.t

Συνολικά χρειάζονται 10 αναγνώσεις από το δίσκο.

3 Φροντιστήρια 2014

Άσκηση 1 ✓

Το ακόλουθο πρόγραμμα είναι αποδοτικό ή όχι:

```
for (int j=0; j<64; j++)
    for (int i=0; i<64;i++)
        x[i][j]=0;
```

Απάντηση

Όχι γιατί ο πίνακας αποθηκεύεται εξορισμού κατά γραμμές. Η βελτιστοποίηση γίνεται ως εξής:

```
for (int i=0; i<64; i++)
    for (int j=0; j<64; j++)
        x[i][j]=0;
```

Άσκηση 2 ✓

data stack text

Ένας υπολογιστής παρέχει σε κάθε διεργασία χώρο διευθύνσεων 65536 bytes που διαιρείται σε σελίδων των 4096 bytes. Ένα συγκεκριμένο πρόγραμμα έχει ένα κείμενο μεγέθους 32768 bytes, δεδομένα μεγέθους 16386 bytes και μια στοίβα μεγέθους 15870 bytes. Θα μπορέσει αυτό το πρόγραμμα να χωρέσει στην μνήμη; Αν το μέγεθος της σελίδας ήταν 512 bytes, θα ταιριάζει; Θυμηθείτε ότι μια σελίδα δεν μπορεί να περιέχει κομμάτια δύο διαφορετικών τμημάτων.

Απάντηση

Για σελίδες των 4096 bytes, χρειάζονται:

- Κείμενο: 32768 / 4096 = 8 σελίδες
- Δεδομένα: 16386 / 4096 = 4,000488281 = 5 σελίδες (για το λίγο ακόμη που χρειάζεται παίρνει ολόκληρη την επιπλέον σελίδα)
- Στοίβα: 15870 / 4096 = 3,874511719 = 4 σελίδες

Συνολικά χρειάζεται 17 σελίδες. Στη μνήμη είναι διαθέσιμες 65536 / 4096 = 16 σελίδες. Άρα το πρόγραμμα δεν χωράει.

Για σελίδες των 512 bytes, χρειάζονται:

- Κείμενο: 32768 / 512 = 64 σελίδες
- Δεδομένα: 16386 / 512 = 32,00390625 = 33 σελίδες
- Στοίβα: 15870 / 512 = 30,99609375 = 31 σελίδες

Συνολικά χρειάζεται 128 σελίδες. Στη μνήμη είναι διαθέσιμες 65536 / 512 = 128 σελίδες. Άρα το πρόγραμμα χωράει.

Παρατήρηση

Σε κάθε διαίρεση παίρνουμε το επάνω ακέραιο μέρος.

Άσκηση 3

TLB

Ο χρόνος αναζήτησης στη μνήμη είναι T_1=5 nsec. Ο χρόνος αναζήτησης στη συσχετιστική μνήμη είναι T_2=1 nsec. Τι ποσοστό επιτυχίας πρέπει να υπάρχει στη συσχετιστική μνήμη ώστε ο συνολικός χρόνος προσπέλασης να είναι 2 nsec;

Απάντηση

$T_{\text{ολικό}}$= Ποσοστό επιτυχίας στη συσχετιστική μνήμη • Χρόνος προσπέλασης στη συσχετιστική μνήμη + Ποσοστό επιτυχίας στη κύρια μνήμη •Χρόνος προσπέλασης στη κύρια μνήμη⇒ 2 nsec=h • 1 nsec + (1-h) •5 nsec⇒h=0,75

Άσκηση 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;

Απάντηση

- 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) στο TLB : 50ns + 400ns + 400ns = 850ns
- Πραγματικός χρόνος προσπέλασης με απλή σελιδοποίηση και TLB :
  - 450*80% + 850*20% = 530ns
- Speed-up = 800/530 = 1.51

Άσκηση 5

Ένας Η/Υ του οποίου οι διεργασίες έχουν 1024 σελίδες στο χώρο διευθύνσεων φυλά τον πίνακα σελίδων του στην κύρια μνήμη. Το overhead που απαιτείται για το διάβασμα μιας λέξης από τον πίνακα σελίδων είναι 5 nsec. Για τη μείωση του overhead ο Η/Υ χρησιμοποιεί μια TLB με 32 ζεύγη (εικονική σελίδα, φυσικό πλαίσιο) και εκτελεί μια αναζήτηση σε 1 nsec. Ποιο hit rate απαιτείται για να μειώσει το μέσο overhead στα 2 nsec;

Απάντηση

Ο τύπος που δίνει το μέσο χρόνο επίδοσης στη συσχετιστική μνήμη (TBL) δίνεται από τον τύπο:

$$t_\mu = \frac{h}{100}\cdot t_c + \left(1-\frac{h}{100}\right)\cdot(t_c + t_p)$$

t_μ: Χρόνος μέσης επίδοσης
t_p: Χρόνος προσπέλασης στον πίνακα σελίδων
t_c: Χρόνος προσπέλασης στη συσχετιστική μνήμη (TBL)
h: Ποσοστό επιτυχίας

Εδώ έχουμε: t_p=5nsec, t_c=1nsec, t_μ=2nsec Κάνουμε αντικατάσταση:

$$2 = \frac{h}{100}\cdot 1 + \left(1-\frac{h}{100}\right)\cdot 6 \Rightarrow 2 = 0.01h + 6 - 0.06h \Rightarrow 0.05h = 4 \Rightarrow h = 80$$

Άρα το ποσοστό επιτυχίας πρέπει να είναι 80%.

Άσκηση 6

Έχουμε εικονικές διευθύνσεις 32 bit και μέγεθος σελίδας 4 KB. Πόσες καταχωρήσεις έχει ο πίνακας σελίδων;

Απάντηση

Το συνολικό μέγεθος του εικονικού χώρου διευθύνσεων (μέγεθος πίνακα σελίδων) είναι 2^{32}. Το μέγεθος σελίδας είναι 4 KB=4096 bytes=2^{12}. Οι καταχωρήσεις (σελίδες) στον πίνακα σελίδων θα είναι $2^{32}/2^{12}=2^{20}$

Άσκηση 7

Έχουμε εικονικές διευθύνσεις 48 bit και φυσικές διευθύνσεις 32 bit και πίνακα σελίδων 1 επιπέδου. Το μέγεθος σελίδας είναι 8 KB. Πόσες καταχωρήσεις έχει ο πίνακας σελίδων;

Απάντηση

Ο συνολικός αριθμός διευθύνσεων είναι 2^{48}. Το μέγεθος σελίδας είναι 8 KB=8192 bytes=2^{13}
Οι καταχωρήσεις στον πίνακα σελίδων θα είναι $2^{48}/2^{13}=2^{35}$

Άσκηση 8

Δίνονται 3 φυσικά πλαίσια μνήμης. Εκτελέστε τον αλγόριθμο LFD (Longest Forward Distance) στην ακολουθία σελίδων:

1 2 3 4 5 2 3 4 1 5 3 4 1 5 4

Απάντηση

Αρχικά	**1**	**2**	**3**	**4**	**5**	**2**	**3**	**4**	**1**	**5**	**3**	**4**	**1**	**5**	**4**
	1	1	1	4	5	5	5	5	5	5	5	5	5	5	5
		2	2	2	2	2	2	4	1	1	1	1	1	1	1
			3	3	3	3	3	3	3	3	3	4	4	4	4
	F	**F**	**F**	**F**	**F**	**H**	**H**	**F**	**F**	**H**	**H**	**F**	**H**	**H**	**H**

Προσοχή

Ο αλγόριθμος LFD (Longest Forward Distance) είναι ο βέλτιστος (OPTIMAL) δηλ. διώχνει τη σελίδα που θα χρησιμοποιηθεί όσο το δυνατόν πιο αργά στο μέλλον

Άσκηση 9

Δώστε παράδειγμα όπου ο LRU είναι καλύτερος από το FIFO και το αντίθετο

Απάντηση

α)Παράδειγμα όπου ο FIFO είναι καλύτερος από τον LRU (δηλ. έχει λιγότερα σφάλματα σελίδας)

FIFO=7 Σφάλματα Σελίδας

Σελίδες	**1**	**2**	**3**	**4**	**5**	**4**	**2**	**1**	**5**
	1	1	1	4	4	4	4	1	1
		2	2	2	5	5	5	5	5
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

LRU= Σφάλματα Σελίδας

Σελίδες	1	2	3	4	5	4	2	1	5
	1	1	1	4	4	4	4	4	5
		2	2	2	5	5	5	1	1
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

b)Παράδειγμα όπου ο LRU είναι καλύτερος από τον FIFO (δηλ. έχει λιγότερα σφάλματα σελίδας)

FIFO= 8 Σφάλματα Σελίδας

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	3	1
		2	2	2	5	5	5	5	4
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

LRU= 7 Σφάλματα Σελίδας

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	4	4
		2	2	2	5	5	5	3	3
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

Άσκηση 10

Η παρακολούθηση του ελεύθερου χώρου στο δίσκο μπορεί να γίνει με τη χρήση μιας λίστας ελεύθερων μπλοκ ή με ένα χάρτη bit (bitmap). Οι διευθύνσεις του δίσκου απαιτούν Δ bit. Αν ένας δίσκος έχει χωρητικότητα Μ μπλοκ από τα οποία Ε είναι ελεύθερα, **δώστε τη συνθήκη με την οποία η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο από χάρτη bit**. Αν το Δ έχει τιμή 16bit εκφράστε την απάντηση σας ως ποσοστό του χώρου δίσκου ο οποίος πρέπει να είναι ελεύθερος.

Απάντηση

Στο χάρτη ψηφίων (bitmap) χρησιμοποιείται 1 bit για κάθε μπλοκ το οποίο υποδηλώνει την κατάσταση του μπλοκ (συγκεκριμένα έχει την τιμή 1 αν το μπλοκ είναι κενό ή 0 αν το μπλοκ είναι γεμάτο). Επειδή ο δίσκος έχει Μ μπλοκ άρα το bitmap έχει μέγεθος Μ bits. Η λίστα ελεύθερων μπλοκ έχει μέγεθος Δ•Ε bits γιατί παρακολουθούμε μόνο τα κενά μπλοκ τα οποία σε πλήθος είναι Ε (σε κάθε ειδικό μπλοκ θεωρούμε ότι βρίσκεται μόνο ένας αριθμός κενού μπλοκ). Η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο από το χάρτη bit εφόσον Δ•Ε<Μ. Εναλλακτικά η λίστα ελεύθερων μπλοκ χρησιμοποιεί λιγότερο χώρο σε σχέση με το χάρτη ψηφίων εφόσον Ε/Μ<1/Δ όπου το Ε/Μ είναι το ποσοστό κενών μπλοκ. Αν Δ=16 bit τότε Ε/Μ<1/Δ=0,0625 ή αν 6,25% του χώρου του σκληρού δίσκου είναι κενός.

Παρατήρηση: Το αποτέλεσμα στο οποίο καταλήγουμε επιβεβαιώνει και τη θεωρία η οποία λέει ότι η λίστα ελεύθερων μπλοκ προτιμάται από το χάρτη ψηφίων όταν ο δίσκος έχει πολύ λίγα κενά μπλοκ.

Άσκηση 11

Βρείτε το βέλτιστο μέγεθος σελίδας έτσι ώστε να ελαχιστοποιείται α)η εσωτερική κατάτμηση (εσωτερικός κατακερματισμός) και β)ο χώρος αποθήκευσης του πίνακα σελίδων

Απάντηση

Έστω ότι το μέσο μέγεθος διεργασίας είναι **μ byte** και το μέγεθος σελίδας είναι **σ byte**. Επιπλέον ας υποθέσουμε ότι κάθε καταχώριση σελίδας απαιτεί **κ bytes**. Ο αριθμός των σελίδων που χρειάζονται για κάθε διεργασία είναι κατά προσέγγιση **μ/σ** και ο χώρος που θα καταληφθεί στον πίνακα σελίδων θα είναι **μκ/σ bytes**. Η μνήμη που χάνεται στην τελευταία σελίδα λόγω της εσωτερικής κατάτμησης είναι **σ/2**. Επομένως η συνολική επιβάρυνση λόγω του πίνακα σελίδων και της εσωτερικής κατάτμησης δίνεται από το άθροισμα δύο όρων:

Επιβάρυνση = μκ/σ +σ/2

Ο 1ος όρος **μκ/σ** (το μέγεθος του πίνακα σελίδων) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μικρό. Ο 2ος όρος **σ/2** (η εσωτερική κατάτμηση) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μεγάλο. Το βέλτιστο μέγεθος πρέπει να βρίσκεται κάπου ανάμεσα. Παίρνουμε την 1η παράγωγο **ως προς σ** και εξισώνουμε με το μηδέν οπότε προκύπτει η εξίσωση:

-μκ/σ^2 +1/2=0

Από την εξίσωση αυτή μπορούμε να κατασκευάσουμε ένα τύπο που δίνει το βέλτιστο μέγεθος σελίδας (λαμβάνοντας υπόψη μόνο τη μνήμη που χάνεται από την κατάτμηση και το μέγεθος του πίνακα σελίδων). Το αποτέλεσμα είναι:

$\sigma = \sqrt{2\mu\kappa}$

Παράδειγμα

Για μ=1 Mbyte και k=8 byte για κάθε καταχώριση στον πίνακα σελίδων, το βέλτιστο μέγεθος σελίδας είναι 4 KBytes

Άσκηση 12

Ο αλγόριθμος LOOK μοιάζει πολύ με τον αλγόριθμο χρονοπρογραμματισμού SCAN. Και οι δύο αλγόριθμοι μετακινούν την κεφαλή του δίσκου προς μια κατεύθυνση την φορά ενώ δίνουν τη δυνατότητα στην κεφαλή να κινηθεί και προς αύξοντα αριθμό τχνών αλλά και προς φθίνοντα αριθμό τχνών. Σε αντίθεση με τον SCAN ο αλγόριθμος LOOK δεν θα φτάσει μέχρι και το τελευταίο ίχνος του δίσκου αλλά μόλις εξυπηρετήσει την τελευταία αίτηση προς μια συγκεκριμένη κατεύθυνση αλλάζει κατεύθυνση για να εξυπηρετήσει και τις υπόλοιπες αιτήσεις.

Μια παραλλαγή του LOOK είναι ο αλγόριθμος C-LOOK ο οποίος κινεί την κεφαλή του σκληρού δίσκου προς μια μόνο κατεύθυνση. Ο αλγόριθμος αυτός ξεκινάει με μια δεδομένη φορά πχ από τον μικρότερο αριθμό ίχνους προς το μεγαλύτερο αριθμό ίχνους και μόλις εξυπηρετήσει και την τελευταία αίτηση επιστρέφει πάλι στην αρχή από όπου ξεκίνησε.

Παράδειγμα

Έστω σκληρός δίσκος με 200 ίχνη (0-199) ο οποίος έχει σε μια δεδομένη στιγμή τις παρακάτω αιτήσεις: ίχνος 11, ίχνος 5, ίχνος 53, ίχνος 88, ίχνος 76, ίχνος 2.

Έστω ότι η κεφαλή τη δεδομένη στιγμή είναι στο ίχνος 30 και κινείται προς αύξοντα αριθμό ίχνους.

LOOK

Ο αλγόριθμος LOOK θα εξυπηρετήσει τις παραπάνω αιτήσεις με την εξής σειρά:

30-->53-->76-->88-->11-->5-->2

C-LOOK

Ο αλγόριθμος C-LOOK θα εξυπηρετήσει τις παραπάνω αιτήσεις με την εξής σειρά:

30-->53-->76-->88-->2-->5-->11

Ο αλγόριθμος LOOK έχει καλύτερο μέσο χρόνο αναζήτησης ίχνους σε σχέση με την SCAN διότι δεν επισκέπτεται τα ακραία ίχνη του δίσκου εκτός και αν υπάρχουν αιτήσεις σχετικές με αυτά

SCAN

Ο αλγόριθμος SCAN θα εξυπηρετήσει τις παραπάνω αιτήσεις με την εξής σειρά:

30-->53-->76-->88-->199-->11-->5-->2

C- SCAN

Ο αλγόριθμος C- SCAN θα εξυπηρετήσει τις παραπάνω αιτήσεις με την εξής σειρά:

30-->53-->76-->88-->199-->0->2-->5-->11

Άσκηση 13

Έστω σκληρός δίσκος με 200 ίχνη (0-199) ο οποίος έχει σε μια δεδομένη στιγμή τις παρακάτω αιτήσεις: 95, 180, 34, 119, 11, 123, 62, 64. Να υπολογιστεί το συνολικό κόστος για τους αλγόριθμους **FIFO/FCFS, SSF, SCAN, LOOK, C-SCAN, C-LOOK**. Αρχικά η κεφαλή είναι στο ίχνος 50.

Απάντηση

FIFO/FCFS

50→95	95→180	180→34	34→119	119→11	11→123	123→62	62→64	
45	**85**	**146**	**85**	**108**	**112**	**61**	**2**	**Σύνολο 644**

SSF

50→62	62→64	64→34	34→11	11→95	95→119	119→123	123→180	
12	**2**	**30**	**23**	**84**	**24**	**4**	**57**	**Σύνολο 236**

SCAN (πάει προς τα δεξιά)

50→62	62→64	64→95	95→119	119→123	123→180	180→199	199→34	34→11	
12	**2**	**31**	**24**	**4**	**57**	**19**	**165**	**23**	**Σύνολο 337**

SCAN (πάει προς τα αριστερά)

50→34	34→11	11→0	0→62	62→64	64→95	95→119	119→123	123→180	
16	**23**	**11**	**62**	**2**	**31**	**24**	**4**	**57**	**Σύνολο 230**

LOOK (πάει προς τα δεξιά)

50→62	62→64	64→95	95→119	119→123	123→180	180→34	34→11	
12	**2**	**31**	**24**	**4**	**57**	**146**	**23**	**Σύνολο 299**

LOOK (πάει προς τα αριστερά)

50→34	34→11	11→62	62→64	64→95	95→119	119→123	123→180	
24	**23**	**51**	**2**	**31**	**24**	**4**	**54**	**Σύνολο 213**

C-SCAN (πάει προς τα δεξιά)

50→62	62→64	64→95	95→119	119→123	123→180	180→199	199→0	0→11	11→34	
12	**2**	**31**	**24**	**4**	**57**	**19**	**199**	**11**	**23**	**Σύνολο 382**

C-SCAN (πάει προς τα αριστερά)

50→34	34→11	11→0	0→199	199→180	180→123	123→119	119→95	95→64	64→62	
16	**23**	**11**	**199**	**19**	**57**	**4**	**24**	**31**	**2**	**Σύνολο 386**

C-LOOK (πάει προς τα δεξιά)

50→62	62→64	64→95	95→119	119→123	123→180	180→11	11→34	
12	**2**	**31**	**24**	**4**	**57**	**169**	**23**	**Σύνολο 322**

C-LOOK (πάει προς τα αριστερά)

50→34	34→11	11→180	180→123	123→119	119→95	95→64	64→62	
16	**23**	**159**	**57**	**4**	**24**	**31**	**2**	**Σύνολο 316**

Άσκηση 14

Το παράδοξο του Belady

- Ο αλγόριθμος FIFO ορισμένες φορές δημιουργεί περισσότερα σφάλματα όταν αυξηθεί ο αριθμός των πλαισίων.
- Θεωρείστε το διπλανό παράδειγμα όπου σε μία διεργασία έχουν δοθεί αρχικά 3 πλαίσια και κατόπιν 4. Η διεργασία αυτή έχει 5 σελίδες, στις οποίες κάνει αναφορά με την εξής σειρά: 0→1→2→3→0→1→4→0→1→2→3→4.
- Με 3 πλαίσια ο αριθμός των σφαλμάτων είναι 9 ενώ με 4 είναι 10.
- Η παρατήρηση αυτή έγινε από τον Belady το 1969.

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

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

Άσκηση 15

Ο αλγόριθμος NFU ονομάζεται και αλγόριθμος γήρανσης. Σε κάθε χτύπο ρολογιού προσθέτει το bit αναφοράς κάθε σελίδας από αριστερά στους αντίστοιχους μετρητές. **Οι διαφορές από τον LRU είναι:**
α)Οι μετρητές έχουν πεπερασμένο μέγεθος και έτσι χάνεται πληροφορία
β)Μετρά αν εμφανίστηκε μια σελίδα σε κάποιο κύκλο ρολογιού, όχι πόσο συχνά

Άσκηση 16

Πέντε εργασίες περιμένουν να εκτελεστούν. Οι αναμενόμενοι χρόνοι εκτέλεσης είναι 9, 6, 3, 5 και X. Με ποια σειρά πρέπει να εκτελεστούν ώστε να ελαχιστοποιηθεί ο μέσος χρόνος απόκρισης; Η απάντηση πρέπει να είναι συνάρτηση του X

Απάντηση

Ο σκοπός του SJF είναι να ελαχιστοποιήσει το μέσο χρόνο απόκρισης. Άρα η σειρά εκτέλεσης είναι η εξής:

$0 < X \leq 3$: X, 3, 5, 6, 9.

$3 < X \leq 5$: 3, X, 5, 6, 9.

$5 < X \leq 6$: 3, 5, X, 6, 9.

$6 < X \leq 9$: 3, 5, 6, X, 9.

$X > 9$: 3, 5, 6, 9, X.

Άσκηση 17

Πέντε εργασίες δέσμης Α έως Ε περιμένουν να εκτελεστούν την ίδια περίπου χρονική στιγμή. Οι χρόνοι εκτέλεσης εκτιμώνται σε 10, 6, 2, 4 και 8 sec αντίστοιχα. Οι προτεραιότητες είναι 3, 5, 2, 1 και 4 αντίστοιχα με 5 την υψηλότερη προτεραιότητα. Για καθένα από τους επόμενους αλγορίθμους να υπολογίσετε το μέσο χρόνο διεκπεραίωσης. Αγνοήστε την επιβάρυνση λόγω εναλλαγής των διεργασιών.

(α)εξυπηρέτηση εκ περιτροπής

(β) χρονοπρογραμματισμός προτεραιοτήτων

(γ) εξυπηρέτηση με βάση τη σειρά άφιξης (εκτέλεση με τη σειρά 10, 6 2, 4, 8)

(δ) εξυπηρέτηση με βάση τη μικρότερη διάρκεια

Απάντηση

(α)

Στον round robin στη διάρκεια των 10 πρώτων sec κάθε διεργασία παίρνει το 1/5 της CPU άρα 2 sec. Στο τέλος τών 10 sec τελειώνει η διεργασία C. Στα επόμενα 8 sec κάθε διεργασία από τις 4 που απέμειναν παίρνει το 1/4 της CPU άρα 2 sec. Στο τέλος των 8 sec τελειώνει η διεργασία D. Στα επόμενα 6 sec κάθε μια από τις 3 εναπομείναντες διεργασίες παίρνει το 1/3 της CPU άρα 2 sec. Στο τέλος των 6 sec τελειώνει η διεργασία B. Στα επόμενα 4 sec κάθε διεργασία παίρνει το 1/2 της CPU άρα 2 sec. Στο τέλος των 4 λεπτών τελειώνει η διεργασία E και μετά από 2 λεπτά τελειώνει και η διεργασία A. Οι χρονικές στιγμές τερματισμού για τις 5 διεργασίες είναι 10, 18, 24, 28 και 30 αντίστοιχα δίνοντας ένα μέσο όρο 22 λεπτών.

(β)

Στο χρονοπρογραμματισμό προτεραιοτήτων πρώτα αρχίζει η B. Τερματίζει μετά από 6 λεπτά. Οι υπόλοιπες διεργασίες με βάση την προτεραιότητα τους τερματίζουν στις χρονικές στιγμές 14, 24, 26, και 30 αντίστοιχα δίνοντας ένα μέσο όρο 18,8 λεπτών.

(γ)
Στο χρονοπρογραμματισμό με βάση τη σειρά άφιξης (εκτέλεση με τη σειρά 10, 6 2, 4, 8) τελειώνουν τις χρονικές στιγμές 10, 16, 18, 22 και 30 αντίστοιχα δίνοντας ένα μέσο όρο 19,2 λεπτά.

(δ)
Στην εξυπηρέτηση με βάση τη μικρότερη διάρκεια οι χρονικές στιγμές τερματισμού των διεργασιών είναι οι 2, 6, 12, 20 και 30 αντίστοιχα δίνοντας ένα μέσο όρο 14 λεπτά

Άσκηση 18

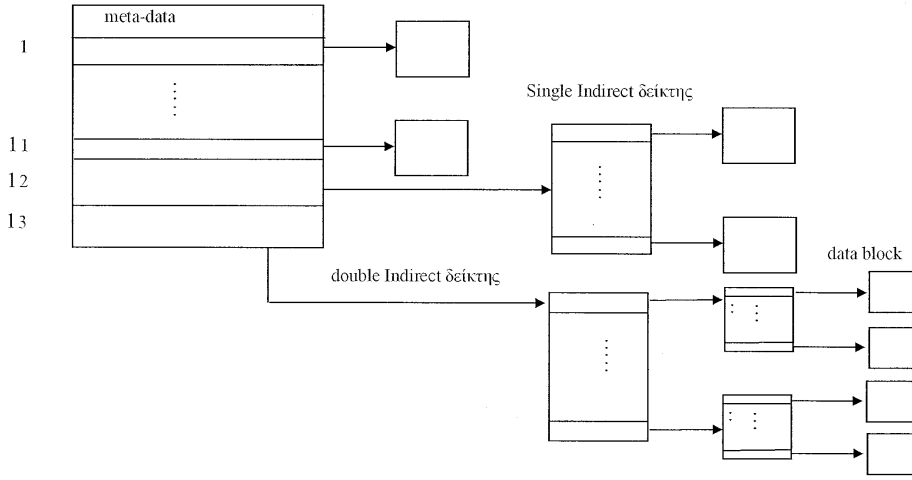
Θεωρείστε ένα file system με μέγεθος block 4.096 bytes και 32-bit δείκτες προς το δίσκο και αρχεία. Κάθε αρχείο έχει 11 direct δείκτες προς block, 1 single indirect δείκτη και 1 double indirect δείκτη.

a) Ποιο το μέγιστο μέγεθος αρχείου που υποστηρίζεται; Ποιο είναι το μέγιστο μέγεθος αρχείου που δεν χρειάζεται double indirect δείκτη;

b) Πόσες προσπελάσεις θα χρειαστούμε στο δίσκο για να διαβάσουμε το τελευταίο byte ενός αρχείου 1 GB (υποθέστε ότι όλα τα δεδομένα που χρειάζεστε βρίσκονται αρχικά στο δίσκο);

c) Έστω ότι έχετε ένα σκληρό δίσκο μεγέθους 1 TB και sectors 4KB και θέλετε να τον αξιοποιήσετε σε ένα σύστημα που το file system του χρησιμοποιεί δείκτες σε block μεγέθους 16-bit. Μπορείτε να βγάλετε συμπέρασμα αν θα δουλέψει/αναγνωριστεί ο δίσκος;

Απάντηση



a)Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 11 direct δείκτες είναι:

- #direct pointers * block size = 11 * 4096 = 45.056 bytes.
- Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 single indirect δείκτη είναι: $\frac{4096}{4} * 4096 = 4.194.304 \text{ bytes}$
- Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 1 double indirect δείκτη είναι $\left(\frac{4096}{4}\right)^2 * 4096 = 4.294.967.296 \text{ bytes}$
- Το μέγιστο μέγεθος αρχείου είναι: 45.056 + 4.194.304 + 4.294.967.269 = 4.299.166.156 ≈ 4 GB
- Το μέγιστο μέγεθος αρχείου χωρίς double indirect δείκτη είναι: 4.556 + 4.194.304 ≈ 4 KB

b)
Το αρχείο μεγέθους 1GB=1.024 MB=1.024 x 1.024KB = 1.024 x 1.024 x 1.024 bytes καταλαμβάνει συνολικά: 1024 x 1024 x 1024: 4096 = 262.144 blocks.

Τα πρώτα 11 blocks στα οποία δείχνουν οι 11 πρώτοι direct δείκτες θέλουν 1 προσπέλαση το καθένα, άρα συνολικά 11 προσπελάσεις. Μένουν 262.144 – 11 = 262.133 blocks.

Ο single indirect δείκτης δείχνει σε 4.096/4=1.024 blocks. Άρα τα επόμενα 1024 blocks θέλουν 2 προσπελάσεις
Μένουν 262.133-1.024=261.109 blocks

Ο double indirect δείκτης δείχνει σε $(4096/4)^2$=1.048.576 blocks. Άρα τα υπόλοιπα 261.109 blocks βρίσκονται μέσα στα 1.048.576 blocks άρα θέλουν 3 προσπελάσεις. Άρα το τελευταίο byte του αρχείου που βρίσκεται μέσα στα 261.109 blocks θέλει 3 προσπελάσεις.

c) Το μέγεθος του δίσκου είναι **Αριθμός κυλίνδρων x Αριθμός Sectors/κύλινδρο x Αριθμός μπλοκ/sector x bytes/μπλοκ**

Αφού δεν δίνεται στην εκφώνηση ο αριθμός κυλίνδρων και ο αριθμός των Sectors/κύλινδρο δεν μπορούμε να βγάλουμε συμπέρασμα για το αν θα δουλέψει ο δίσκος.

Άσκηση 19

Υποθέστε ότι έχουμε μία μνήμη με σελιδοποίηση κατ' απαίτηση. Ο πίνακας σελίδων διατηρείται σε καταχωρητές. Η εξυπηρέτηση ενός σφάλματος σελίδας διαρκεί 8 ms αν είναι διαθέσιμο ένα κενό πλαίσιο ή αν η σελίδα που αντικαθίσταται δεν είναι τροποποιημένη και 20 ms αν είναι τροποποιημένη. Ο χρόνος της προσπέλασης της μνήμης είναι 100 ns.
Υποθέστε ότι η σελίδα που πρόκειται να αντικατασταθεί είναι τροποποιημένη το 70% των φορών. Ποιος είναι ο μέγιστος αποδεκτός ρυθμός σφαλμάτων σελίδας για πραγματικό χρόνο πρόσβασης όχι μεγαλύτερο από 200 ns;

Απάντηση

$200ns \geq (1-p) \times 100ns + (0,3 \times p) \times 8ms + (0,7 \times p) \times 20ms$
Από την παραπάνω ανίσωση προκύπτει ότι $p \leq 0,000006$

Άσκηση 20

Θεωρήστε ένα σύστημα σελιδοποίησης κατ' απαίτηση με ένα δίσκο σελιδοποιημένο που έχει μέσο χρόνο προσπέλασης και μεταφοράς 20 ms. Οι διευθύνσεις μεταφράζονται μέσω ενός πίνακα σελίδων στην κύρια μνήμη, με χρόνο προσπέλασης 1 μs ανά προσπέλαση μνήμης. Έτσι, κάθε αναφορά μνήμης μέσω του πίνακα σελίδων απαιτεί δύο προσπελάσεις. Για να βελτιωθεί αυτός ο χρόνος, έχουμε προσθέσει μία συσχετιστική μνήμη που μειώνει το χρόνο προσπέλασης σε μία αναφορά μνήμης αν η εγγραφή στον πίνακα σελίδων βρίσκεται στη συσχετιστική μνήμη.
Υποθέστε ότι 80% των προσπελάσεων βρίσκονται στη συσχετιστική μνήμη και ότι, από τις υπόλοιπες, 10% (ή 2% του συνόλου) προσκαλεί σφάλματα σελίδας. Ποιος είναι ο πραγματικός χρόνος πρόσβασης;

Απάντηση

Χρόνος πρόσβασης = $0,8 \times 1\mu s + 0.18 \times (1\mu s + 1\mu s) + 0.02 \times (1\mu s + 20000\mu s + 1\mu s) =$
$0,8 + 0,36 + 400.04 = 401,20\mu s$

Άσκηση 21
Έστω δ διεργασίες κάθε διεργασία θέλει τα πολύ μ πόρους και υπάρχουν π πόροι διαθέσιμοι. Ποια η συνθήκη για να μην υπάρχει αδιέξοδο;
Απάντηση
Η χειρότερη περίπτωση είναι ότι όλες οι διεργασίες έχουν μ-1 πόρους και ζητούν ακόμα 1. Άρα $\pi \geq \delta(\mu-1)+1$.

Άσκηση 22
Ποια από τις επόμενες λειτουργίες μπορεί να εκτελεστεί μόνο σε κατάσταση λειτουργίας πυρήνα
Απάντηση
α)Απενεργοποίηση Διακοπών
β)Ανάγνωση ρολογιού

γ)Απόδοση τιμής στο ρολόι
δ)Αλλαγή memory map

Απάντηση

α)
γ)
δ)

Άσκηση 23

Για καθένα από τα παρακάτω system calls δώστε παράδειγμα ώστε να μην εκτελεστεί επιτυχώς

- fork→αν ξεπεραστεί ο μέγιστος αριθμός διεργασιών που μπορεί να έχει το σύστημα
- exec(fd)→όταν δεν υπάρχει το fd ή όταν δεν έχουμε δικαιώματα εκτέλεσης ή όταν το αρχείο δεν είναι εκτελέσιμο
- unlink(fd)→ όταν δεν υπάρχει το fd ή όταν δεν έχουμε δικαιώματα εκτέλεσης

Άσκηση 24

Στον παρακάτω πίνακα σελίδων ποια σελίδα θα αντικατασταθεί με βάση τον αλγόριθμο Working Set αν τ=400 και ο τρέχον εικονικός χρόνος είναι 2204;

Σελίδα	Χρόνος τελευταίας προσπέλασης	R bit
0	2084	1
1	2003	1
2	1980	1
3	1213	0
4	2014	1
5	2020	1
6	2032	1
7	1620	0

Απάντηση

Ο αλγόριθμος αφαιρεί σελίδα η οποία έχει R=0 και ηλικία > τ (ηλικία = Τρέχον εικονικός χρόνος - Χρόνος τελευταίας προσπέλασης).

Η σελίδα με τον μικρότερο χρόνο και R=0 είναι η 3. Η ηλικία της είναι 2204-1213=991>400 άρα αφαιρείται.

Άσκηση 25

Γιατί ο πίνακας διεργασιών είναι απαραίτητος στα συστήματα χρονομερισμού. Είναι επίσης απαραίτητος σε ένα σύστημα προσωπικού Η/Υ που εκτελεί μόνο 1 διεργασία η οποία καταλαμβάνει όλη τη μηχανή;

Απάντηση

Είναι απαραίτητος γιατί όταν εξαντλείται το κβάντο χρόνου μιας διεργασίας θα μπει στον επεξεργαστή η επόμενη και πρέπει να γνωρίζουμε την κατάσταση της πρώτης δηλ. σε ποιο σημείο είχε φτάσει ώστε όταν ξαναεκτελεστεί να ξέρουμε από ποιο σημείο θα συνεχίσει. Σε προσωπικό Η/Υ δεν είναι απαραίτητος.

Άσκηση 26

Ένας υπολογιστής έχει τέσσερα πλαίσια σελίδας. Ο χρόνος φόρτωσης και τελευταίας προσπέλασης καθώς και οι τιμές των R και M bits. Δίνονται παρακάτω.

Σελίδα	Φόρτωση	Προσπέλαση	R	M
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	110	285	1	1

Ποια σελίδα θα αντικατασταθεί σύμφωνα με τους αλγόριθμους:

a. NRU
b. LRU
c. FIFO
d. Δεύτερης ευκαιρίας

Απάντηση

a. NRU: Αντικαθιστά σελίδα από την κατηγορία 0 (R=0, M=0), άρα τη σελίδα 2
b. FIFO: Με βάση την φόρτωση θα αφαιρέσει εκείνη με το μικρότερο χρόνο, δηλαδή τη σελίδα 3
c. LRU: Με βάση την τελευταία προσπέλαση θα αφαιρέσει εκείνη με το μικρότερο χρόνο, δηλαδή, τη σελίδα 1
d. Δεύτερης ευκαιρίας: Με βάση την φόρτωση, υποψήφια είναι η σελίδα 3. Έχει, όμως, R=1, οπότε γίνεται R=0 και βρίσκει την αμέσως επόμενη που είναι η 0. Και αυτή έχει R=1, το οποίο γίνεται 0 και βρίσκει την επόμενη η οποία είναι η 2. Αυτή έχει R=0 οπότε και αφαιρείται.

Άσκηση 27

Τι γνωρίζετε για τον κανόνα του 50%

Απάντηση

Ο αριθμός των διεργασιών είναι περίπου διπλάσιος από τον αριθμό των κενών. Ισχύει γιατί 2 γειτονικά κενά συγχωνεύονται σε ένα, ενώ 2 γειτονικές διεργασίες δεν συγχωνεύονται.

Έστω m ο χώρος μνήμης, s το μέσο μέγεθος διεργασίας, k είναι το πλήθος των οπών και k•s το μέσο μέγεθος οπών και n το πλήθος των διεργασιών. Ισχύει ότι: $m = n \bullet s + \frac{n}{2} \bullet k \bullet s$ όπου $\frac{n}{2}$ είναι το πλήθος των οπών λόγω του κανόνα του 50%. **Το ποσοστό της μη χρησιμοποιούμενης μνήμης** είναι: $\text{Ποσοστό\_Οπών} = \frac{\text{μέγεθος οπών}}{\text{μέγεθος μνήμης (m)}} = \frac{\frac{n}{2} \bullet ks}{ns + \frac{n}{2} \bullet ks} = \frac{k}{k+2}$

Άσκηση 28

Δίνονται οι ακόλουθες διεργασίες με τις προτεραιότητες και τη διάρκεια που φαίνεται στον πίνακα:

Διεργασία	Διάρκεια	Προτεραιότητα
Π1	1	5
Π2	2	4
Π3	2	3
Π4	5	1
Π5	15	2

Το κβάντο χρόνου (time slice) είναι 1 msec. Να εφαρμοστούν οι αλγόριθμοι

a)Round-Robin (Εκ Περιτροπής)

b)Round-Robin (Εκ Περιτροπής) με προτεραιότητες

c)Shortest Job First

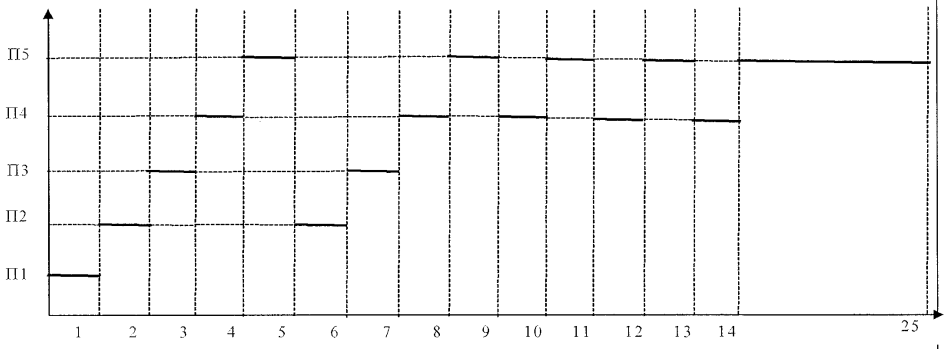
και να υπολογιστούν για κάθε αλγόριθμο

α)Ο μέσος χρόνος απόκρισης

β)Ο μέσος χρόνος αναμονής

Απάντηση

a)Round-Robin



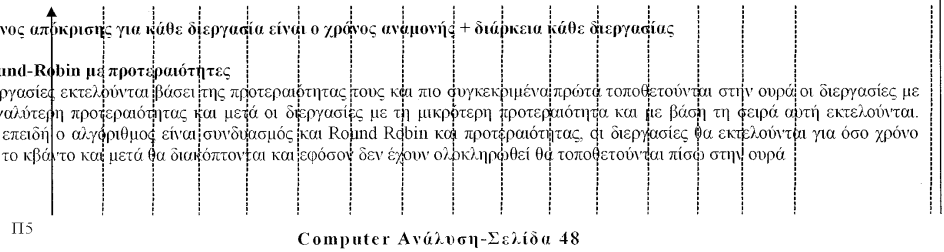
$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+4+5+9+10}{5} = 5{,}6$$

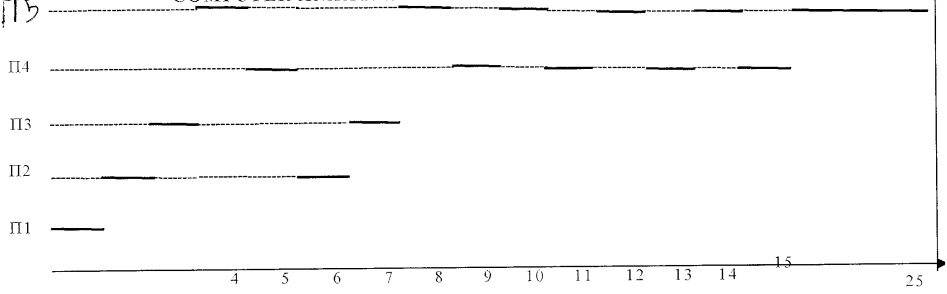
$$\text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(4+2)+(5+2)+(9+5)+(10+15)}{5}$$

Ο χρόνος απόκρισης για κάθε διεργασία είναι ο χρόνος αναμονής + διάρκεια κάθε διεργασίας

b) Round-Robin με προτεραιότητες

Οι διεργασίες εκτελούνται βάσει της προτεραιότητας τους και πιο συγκεκριμένα πρώτα τοποθετούνται στην ουρά οι διεργασίες με τη μεγαλύτερη προτεραιότητας και μετά οι διεργασίες με τη μικρότερη προτεραιότητα και με βάση τη σειρά αυτή εκτελούνται. Τώρα επειδή ο αλγόριθμος είναι συνδυασμός και Round Robin και προτεραιότητας, οι διεργασίες θα εκτελούνται για όσο χρόνο ορίζει το κβάντο και μετά θα διακόπτονται και εφόσον δεν έχουν ολοκληρωθεί θα τοποθετούνται πίσω στην ουρά



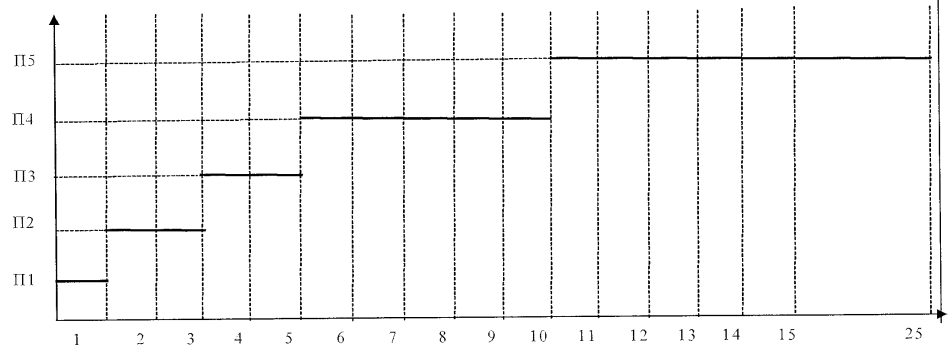


$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+4+5+10+10}{5} = 5{,}8$$

$$\text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(4+2)+(5+2)+(10+5)+(10+15)}{5}$$

c) SJF

Οι διεργασίες εκτελούνται βάσει της διάρκειας τους και πιο συγκεκριμένα πρώτα τοποθετούνται στην ουρά οι διεργασίες με τη μικρότερη διάρκεια και μετά οι πιο χρονοβόρες διεργασίες και μετά εκτελούνται με βάσει αυτή τη σειρά. Εδώ δεν υπάρχει κβάντο χρόνου οπότε οι διεργασίες θα εκτελούνται πλήρως και δεν θα διακόπτονται.



$$\text{Μέσος Χρόνος Αναμονής} = \frac{0+1+3+5+10}{5} = 3{,}8$$

$$\text{Μέσος Χρόνος Απόκρισης} = \frac{(0+1)+(1+2)+(3+2)+(5+5)+(10+15)}{5}$$

Άσκηση 29

Σε όλους τους Η/Υ τουλάχιστον ένα τμήμα των διακοπών είναι γραμμένο σε συμβολική γλώσσα γιατί:

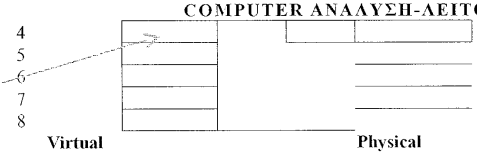
Απάντηση

Για να γίνει πολύ γρήγορα (μέσω hardware) σε πολύ μικρότερο χρόνο από όσο διαρκεί ένας κύκλος CPU.Επίσης θέλουμε να είναι αξιόπιστες και δεν εμπιστευόμαστε τον compiler αλλά εμείς τι ακριβώς να κάνει.

Άσκηση 30

Δίνεται η εντολή MOV REG, 1024. Να βρεθεί η φυσική διεύθυνση στην οποία θα απεικονιστεί

0-4191			0-4.191
4192-8191			4.192-8.191
8192-16383			8.192-16.383



Απάντηση

Η εικονική διεύθυνση 1024 βρίσκεται στην 1η σελίδα σε απόσταση 1024 από την αρχή της (offset). Προσθέτουμε το ίδιο offset στην αρχή του φυσικού πλαισίου στο οποίο απεικονίζεται η σελίδα δηλ. προσθέτουμε το 1024 στη διεύθυνση 4192 και προκύπτει ότι 4192+1024=5216

Άσκηση 31

Πως γίνεται η ανίχνευση αδιεξόδου με πολλαπλούς πόρους κάθε τύπου;

Απάντηση

Ελέγχω κάθε γραμμή του πίνακα των πόρων αν είναι διαθέσιμοι οι πόροι που ζητούνται. Αν αυτό δεν ισχύει για καμία γραμμή τότε έχουμε αδιέξοδο δηλ. $O(n^2m)$.

Άσκηση 32

Έστω PC με 1 MB μνήμη και 200K καταλαμβάνει το λειτουργικό σύστημα και 200K χρησιμοποιεί μια μέση διεργασία. Άρα στη μνήμη χωράνε συνολικά 4 διεργασίες όπως δείχνει το ακόλουθο σχήμα:

ΔΣ	Δ1	Δ2	Δ3	Δ4
200K	200	200	200	200

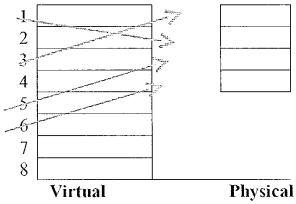
Η πιθανότητα η CPU να μην είναι απασχολημένη είναι 80 %. Η αξιοποίηση της CPU (CPU utilization) είναι $1-0,80^4=59\%$. Αν προσθέσουμε 1 MB επιπλέον μνήμης τότε θα έχουμε συνολικά 9 διεργασίες όπως δείχνει το ακόλουθο σχήμα

ΔΣ	Δ1	Δ2	Δ3	Δ4	Δ5	Δ6	Δ7	Δ8	Δ9
200K	200	200	200	200	200	200	200	200	200

Άρα το CPU utilization είναι τώρα $1-0,80^9=87\%$. Όσο περισσότερες διεργασίες είναι φορτωμένες στην κύρια μνήμη δηλ. όσο περισσότερες διεργασίες εκτελούνται παράλληλα τόσο περισσότερα αυξάνεται το CPU utilization.

Άσκηση 33

Θεωρείστε δύο διεργασίες Δ1 και Δ2 που τρέχουν σε ένα σύστημα με σελίδες/πλαίσια μεγέθους 8 KB και μέγεθος εικονικής μνήμης 64 KB. Το σχήμα απεικονίζει το χάρτη μνήμης (μέρος του ΠΣ) για τη Δ1. Απαντήστε στις επόμενες 3 ερωτήσεις



1. Μια αναφορά της Δ1 σε ποια από τις παρακάτω εικονικές διευθύνσεις θα προκαλέσει σφάλμα σελίδας (οι αριθμοί είναι στο δεκαδικό σύστημα)

α. 9.000

β. 16.500

γ.42.000

δ. Όλες οι παραπάνω

Απάντηση

Α/Α Σελίδας	Διευθύνσεις/Σελίδα	Presence Bit
1	0-8.191	1
2	8.192-16.383	0
3	16.384-24.575	1

4	24.576-32.767	0
5	32.768-40.959	1
6	40.960-49.151	1
7	49.152-57.343	0
8	57.344-65.535	0

Μία αναφορά της Δ1 στην εικονική διεύθυνση 9000 θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 9000 ανήκει στη 2η σελίδα και η 2η σελίδα δεν έχει φορτωθεί σε φυσικό πλαίσιο μνήμης**

Μία αναφορά της Δ1 στην εικονική διεύθυνση 16.500 ΔΕΝ θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 16.500 ανήκει στη 3η σελίδα και η 3η σελίδα έχει φορτωθεί ήδη σε φυσικό πλαίσιο μνήμης**

Μία αναφορά της Δ1 στην εικονική διεύθυνση 42.000 ΔΕΝ θα προκαλέσει σφάλμα σελίδας **διότι η εικονική διεύθυνση 42.000 ανήκει στη 6η σελίδα και η 6η σελίδα έχει φορτωθεί ήδη σε φυσικό πλαίσιο μνήμης**

Άρα η σωστή απάντηση είναι η α.

2. Η επόμενη αναφορά της Δ2 σε ποια από τις παρακάτω εικονικές διευθύνσεις θα προκαλέσει σφάλμα σελίδας (οι αριθμοί είναι στο δεκαδικό σύστημα)

α. 9.000

β. 16.500

γ.42.000

δ. Όλες οι παραπάνω

Απάντηση

δ. Όλες οι παραπάνω

Θεωρούμε ότι η Δ2 δεν έχει φορτώσει ακόμα στη φυσική μνήμη τις σελίδες που χρειάζεται οπότε θα προκληθεί σφάλμα σελίδας από όλες τις αναφορές της Δ2 στις προηγούμενες διευθύνσεις

3. Αν η Δ1 διαγράψει τα περιεχόμενα της 1ης 3ης 5η2 και 6ης σελίδας της, ποιες μετέπειτα αναφορές της Δ1 σε αυτές τις σελίδες θα προκαλέσουν σφάλματα σελίδας;

α. Αναφορές στις μονές σελίδες

β. Αναφορές στις ζυγές σελίδες

γ. Αναφορές σε όλες τις σελίδες

Απάντηση

γ. Αναφορές σε όλες τις σελίδες

Αν η Δ1 διαγράψει τα περιεχόμενα της 1, 3, 5, 6 σελίδας, όλες οι μετέπειτα αναφορές της Δ1 σε αυτές τις σελίδες θα προκαλέσουν σφάλματα σελίδας διότι δεν θα απεικονίζεται καμία σελίδα της Δ1 στη φυσική μνήμη

Άσκηση 34

Υποθέστε ότι στη μνήμη μπορούν να βρίσκονται μέχρι 3 σελίδες. Μια διεργασία προσπελαύνει τις παρακάτω σελίδες μνήμης:
7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0
Πόσα page fault θα συμβούν αν χρησιμοποιήσουμε τον αλγόριθμο «Ρολόι» για αντικατάσταση σελίδων δεδομένου ότι όλα τα ψηφία είναι αρχικά R=0

Απάντηση

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0
→	7*	7*	→7*	2*	2*	→2*	→2*	4*	4*	4*	→4	3*	3*	3	→3	0*	0*	0	→0
	→	0*	0*	→0	→0*	0	0*	→0	2*	2*	2	→2	→2*	1*	1*	→1*	→1*	7*	7*
		→	1*	1	1	3*	3*	3	→3	→3*	0*	0*	0*	→0	2*	2*	2*	→2	0*
	F	**F**	**F**	**F**	**H**	**F**	**H**	**F**	**F**	**H**	**F**	**F**	**H**	**F**	**F**	**F**	**H**	**F**	**F**

Άρα τα page faults είναι 14.

Άσκηση 35

Έστω ένα σύστημα με 16 bit εικονικές και φυσικές διευθύνσεις. Το μέγεθος κάθε σελίδας είναι 256 byte και έστω μία διεργασία η οποία περιέχει τα παρακάτω δεδομένα.

1	32	4F
1	1A	C3
1	89	22
0	42	82

Ο πίνακας περιέχει από αριστερά προς τα δεξιά το virtual bit, την εικονική σελίδα και το πλαίσιο σελίδας όλες σε 16δική μορφή.
α. Αν η διεργασία προσπελάσει την εικονική διεύθυνση 1AF2 ποιά θα είναι η φυσική διεύθυνση που θα τροποποιηθεί;
β. Αν η διεργασία προσπελάσει τη φυσική διεύθυνση 2222 ποιά θα είναι η εικονική διεύθυνση που θα τροποποιηθεί;

Απάντηση

1	32	4F
1	1A	C3
1	89	22
0	42	B2

α) 1AF2 εικονική διεύθυνση σε ποια φυσική διεύθυνση αντιστοιχεί;
Η εικονική διεύθυνση $(1AF2)_{16} = (6898)_{10}$. Ανήκει στη σελίδα 6898 div 256 = 26.94 ≈ 26. Η σελίδα 26 ξεκινάει από τη διεύθυνση 26x256=6656.Άρα το offset δηλαδή η μετατόπιση της διεύθυνσης (1AF2) αυτής από την αρχική διεύθυνση της σελίδας 26 είναι: 6898 – 6656 = 242 offset. Η εικονική σελίδα $(26)_{10} = (1A)_{16}$ αντιστοιχεί (απεικονίζεται) στο φυσικό πλαίσιο μνήμης $(C3)_{16} = (195)_{10}$. Το φυσικό πλαίσιο 195 αρχίζει από τη διεύθυνση 195x256 = 49920 + 242 offset = $(50162)_{10}$ και σε δεκαεξαδική απεικόνιση η φυσική διεύθυνση είναι C3F2.

β) 2222 φυσική διεύθυνση σε ποια εικονική αντιστοιχεί;
Η φυσική διεύθυνση (2222)16 = (8738)10 ανήκει στο φυσικό πλαίσιο 8738 div 256 = 34 (ακέραιο πηλίκο). Το 34ο φυσικό πλαίσιο ξεκινά από τη διεύθυνση 34x256= 8704 άρα το offset από την αρχή της σελίδας είναι:
8738-8704=34. Το φυσικό πλαίσιο $(34)_{10} = (22)_{16}$ αντιστοιχεί στην εικονική σελίδα $(89)_{16} = (137)_{10}$. Η σελίδα 137 αρχίζει από τη διεύθυνση 137x256=35072. Σε αυτή την αρχική διεύθυνση προσθέτουμε το offset που είναι 34 και προκύπτει η εικονική διεύθυνση που είναι 35072+34 =$(35106)_{10}$ = $(8922)_{16}$.

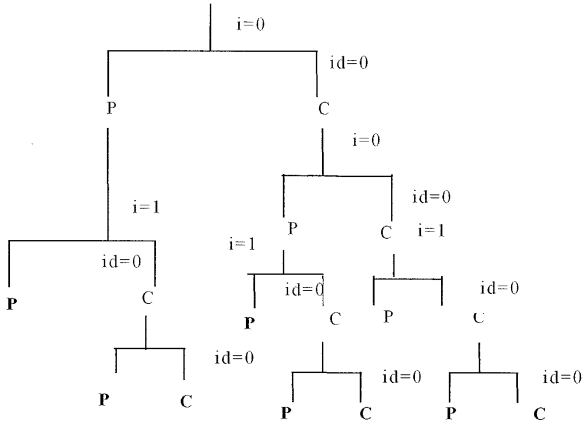
Άσκηση 36

Αν υποθέσουμε πως η μόνη διεργασία σε ένα σύστημα είναι αυτή που εκτελεί τον παρακάτω κώδικα, πόσες διεργασίες θα υπάρχουν συνολικά στο σύστημα μετά την εκτέλεση του κώδικα; Εξηγήστε ακριβώς πως καταλήγετε σε αυτό. Υποθέστε πως στο παράδειγμα μας η fork() επιτυγχάνει πάντοτε (Η fork() επιστρέφει 0 στη διεργασία παιδί και ένα αριθμό διάφορο του 0 στην διεργασία πατέρα)

```
for(i=0; i<2; i++) {
    id=fork();

    if (id==0) {
        fork();
    }
}
```

Απάντηση



Τελικά θα έχουμε 9 διεργασίες (όσα και τα φύλλα του τελικού δέντρου)

4 Πιθανές Ερωτήσεις Θεωρίας

1. Δώστε ένα ορισμό για τις preemptive και non-preemptive πολιτικές δρομολόγησης και περιγράψτε τις διαφορές τους

Απάντηση

Οι αλγόριθμοι χρονοπρογραμματισμού μπορούν να διαιρεθούν σε 2 κατηγορίες σε σχέση με τον τρόπο που αντιμετωπίζουν τις διακοπές ρολογιού. Οι μη προεκτοπιστικοί (non-preemptive) αλγόριθμοι χρονοπρογραμματισμού επιλέγουν μια διεργασία και της επιτρέπουν να εκτελεστεί μέχρι να μπλοκαριστεί (από I/O ή γιατί περιμένει κάποια άλλη διεργασία) ή μέχρι να επιστρέψει εθελοντικά τον έλεγχο της CPU. Συνεπώς στην περίπτωση αυτή δεν λαμβάνονται αποφάσεις χρονοπρογραμματισμού κατά τη διάρκεια των διακοπών ρολογιού.

Αντίθετα οι προεκτοπιστικοί (pre-emptive) αλγόριθμοι χρονοπρογραμματισμού επιλέγουν κάθε φορά τη διεργασία που θα εκτελεστεί στη συνέχεια αλλά δεν της επιτρέπουν να εκτελεστεί περισσότερο από ένα συγκεκριμένο χρονικό διάστημα. Αν η εκτελούμενη διεργασία συνεχίσει να εκτελείται και μετά το πέρας του διαστήματος αυτού, το σύστημα την αναστέλλει υποχρεωτικά και ο χρονοπρογραμματιστής επιλέγει κάποια άλλη διεργασία (αν υπάρχει έστω και μια διαθέσιμη). Για να υλοποιηθεί ο προεκτοπιστικός χρονοπρογραμματισμός απαιτείται να ρυθμιστεί το ρολόι ώστε να προκαλεί διακοπές ακριβώς τη στιγμή που τελειώνει κάθε φορά το προκαθορισμένο χρονικό διάστημα ώστε να επιστρέφει με αυτόν τον τρόπο ο έλεγχος της στο χρονοπρογραμματιστή.

2.Τι ονομάζεται ασφαλής κατάσταση και τι ανασφαλής κατάσταση

Απάντηση

Ένα σύστημα είναι σε ασφαλή κατάσταση όταν υπάρχει τουλάχιστον ένας τρόπος ώστε να ικανοποιήσουμε τις απαιτήσεις όλων των διεργασιών (για πόρους) ακόμα και αν αυτές χρειαστούν τη μέγιστη ανάγκη τους σε πόρους.

Ένα σύστημα είναι σε ανασφαλή κατάσταση όταν οι πόροι δεν επαρκούν για να ικανοποιήσουμε τις ανάγκες των διεργασιών.
Μια ανασφαλής κατάσταση δεν οδηγεί υποχρεωτικά σε αδιέξοδο γιατί α)μια διεργασία μπορεί να μη χρειαστεί τη μέγιστη ανάγκη της σε πόρους και β)μπορεί να αποδεσμεύσει οικειοθελώς κάποιους πόρους και να τους πάρει κάποια άλλη διεργασία και το σύστημα να μην οδηγηθεί σε αδιέξοδο

3.Ποιες είναι οι ικανές και αναγκαίες συνθήκες για να συμβεί αδιέξοδο;

Απάντηση

Για να εμφανιστεί αδιέξοδο πρέπει να ικανοποιούνται ταυτόχρονα 4 συνθήκες:

- mutual exclusion (αμοιβαίος αποκλεισμός): Μόνο μια διεργασία μπορεί να χρησιμοποιεί έναν πόρο.
- **Κατοχή και αναμονή (hold & wait)**: Οι διεργασίες που συμμετέχουν στο αδιέξοδο πρέπει και να κατέχουν κάποιο πόρο αλλά και να περιμένουν για κάποιο πόρο.
- **Μη προεκτοπισμός** (No preemption): Μόνο η κατέχουσα διεργασία μπορεί ν' απελευθερώσει τον πόρο - δηλ. ο πόρος δεν μπορεί ν' αφαιρεθεί από τη διεργασία.
- **κυκλική αναμονή** (circular wait): → Ν διεργασίες, όπου κάθε διεργασία περιμένει για έναν πόρο που τον κατέχει η επόμενη διεργασία στην αλυσίδα.

4.Τι ονομάζεται system call και τι ονομάζεται interrupt και ποιες οι διαφορές τους;

Απάντηση

- Ένα **system call** είναι ένα interface για "συνομιλήσει" ένα πρόγραμμα με το λειτουργικό σύστημα. Όταν ένα πρόγραμμα ζητά μια υπηρεσία (για παράδειγμα το άνοιγμα ενός αρχείου) το ίδιο δεν έχει τα δικαιώματα από τον πυρήνα του λειτουργικού συστήματος τότε χρησιμοποιεί ένα system call. Οι διεργασίες χρηστών δεν έχουν τα ίδια δικαιώματα με τις διεργασίες που αλληλεπιδρούν απευθείας με το λειτουργικό. Για παράδειγμα όταν θέλουμε ένα πρόγραμμα να επικοινωνήσει με εξωτερικές συσκευές τότε θα χρειαστεί να χρησιμοποιήσει system call.

- Κατά τη διάρκεια εκτέλεσης ενός προγράμματος μπορούν να υπάρχουν συμβαντα τα οποία μπορούν να προκαλέσουν το σταμάτημα της CPU. Τέτοια συμβάντα ονομάζονται **interrupts**. Τα interrupts μπορούν να προκαλούνται είτε από σφάλματα υλικού είτε από σφάλματα λογισμικού. Οι διακοπές υλικού (Hardware interrupts) ονομάζονται απλά Interrupts ενώ οι διακοπές λογισμικού ονομάζονται Εξαιρέσεις (Exceptions) or Παγίδες (Traps). Όταν συμβεί διακοπή (είτε υλικού είτε λογισμικού) ο έλεγχος μεταφέρεται σε μια ειδική υπορουτίνα που ονομάζεται ISR (Interrupt Service Routine) η οποία αντιμετωπίζει τη διακοπή.

Η διαφορά μεταξύ System Call και Interrupt είναι η εξής:

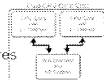
- Το System call είναι μια κλήση σε υπορουτίνα που είναι ενσωματωμένη στο σύστημα ενώ το Interrupt είναι ένα συμβάν το οποίο αναγκάζει τον επεξεργαστή να διακόψει προσωρινά την τρέχουσα εκτέλεση. Μια βασική διαφορά είναι ότι τα system calls είναι σύγχρονα ενώ τα interrupts δεν είναι. Αυτό σημαίνει ότι τα system calls εμφανίζονται σε συγκεκριμένες χρονικές στιγμές (που συνήθως καθορίζονται από τον προγραμματιστή) ενώ τα interrupts μπορούν να συμβούν σε οποιαδήποτε χρονική στιγμή εξαιτίας ενός απροσδόκητου συμβάντος όπως το πάτημα ενός πλήκτρου στο πληκτρολόγιο από ένα χρήστη. Έτσι όταν εμφανιστεί ένα system call ο επεξεργαστής το μόνο που έχει να κάνει είναι να θυμάται το που ακριβώς να επιστρέψει αλλά σε

περίπτωση που εμφανιστεί ένα interrupt τότε ο επεξεργαστής πρέπει να θυμάται και το σημείο που θα επιστρέψει αλλά και την κατάσταση του συστήματος. Σε αντίθεση με ένα system call, ένα interrupt δεν έχει να κάνει κάτι σε σχέση με το πρόγραμμα που εκτελείται.

5. Ποια τα Πλεονεκτήματα των Νημάτων

Απάντηση

1. **Responsiveness (Βελτίωση Χρόνου Απόκρισης Εφαρμογών)**
   - Χρήσιμο σε διαδραστικές εφαρμογές (GUIs), όπου μπορούμε να **«αποκρύψουμε»** μεγάλες **καθυστερήσεις** προερχόμενες από **I/Os**.
   - Σημειώστε ότι και ο ίδιος ο πυρήνας χρησιμοποιεί πολλαπλά threads (π.χ. έλεγχος διαθέσιμης μνήμης).
2. **Resource Sharing (Διαμοιρασμός Πόρων)**
   - Ένα thread **διαμοιράζεται** όλα τα **δεδομένα** (ακόμα και η προσωπική **στοίβα** του είναι προσβάσιμη) με τα υπόλοιπα νήματα της ίδιας διεργασίας.
   - Επομένως **δεν χρειάζεται** κάποιος μηχανισμός **IPC** (shared memory, pipes, fifo, etc) για να μπορούν **δυο νήματα** να **μοιράζονται πόρους** (δομές δεδομένων, μεταβλητές, κτλ)
3. **Economy: Οικονομία Δημιουργίας και Χρονοδρομολόγησης**
   - Το fork() είναι πολύ ακριβό (γενικά η δέσμευση μνήμης είναι ακριβή).
   - Για παράδειγμα στο Solaris η **δημιουργία** μιας διεργασίας είναι **30 φορές πιο αργή** από την δημιουργία ενός νήματος.
   - Επίσης η εναλλαγή διεργασιών (**context switching**) είναι **5 φορές πιο αργή από την «εναλλαγή» νημάτων**
4. **Multiprocessing: Μέγιστη Αξιοποίηση Πολυεπεξεργαστών**
   - Σε συστήματα 1 επεξεργαστή, τα νήματα εκτελούνται **ψευδό-παράλληλα**, ενώ ένα νήμα κάνει block (π.χ., I/O) ένα άλλο εκτελείται.
   - Σε συστήματα N επεξεργαστών, τα νήματα εκτελούνται πραγματικά παράλληλα. Π.χ. Όλα τα λειτουργικά συστήματα υποστηρίζουν σήμερα **SMP (symmetric multiprocessing)**
     - 2 ή περισσότεροι επεξεργαστές σε κοινή μνήμη (π.χ. Intel's Xeon, Core Duo, Intel Quad Core, Intel Itanium, AMD Opteron)
     - Τεχνολογία **Hyper-threading** (SMT): multiple logical cores πάνω από 1 physical core



6. Ποια τα Μειονεκτήματα των Νημάτων

Απάντηση

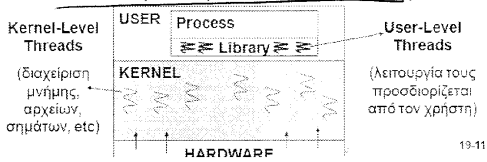
Ανάπτυξη Κώδικα: Η ανάπτυξη κώδικα γίνεται σημαντικά πιο ακριβή λόγω προβλημάτων **συγχρονισμού** και **διαμοιρασμού κοινόχρηστης μνήμης** μεταξύ των νημάτων μιας διεργασίας.

Αποσφαλμάτωση: Γίνεται πιο δύσκολη διότι είναι πιο δύσκολο να ελέγξουμε την ασύγχρονη ροή εκτέλεσης (παρόλο που εργαλεία όπως **gdb** υποστηρίζουν debugging με threads).

7. Ποια είδη νημάτων γνωρίζετε;

Απάντηση

- Υπάρχουν δυο είδη νημάτων
  - User-Level Threads (στο user space)
  - Kernel-Level Threads (στο kernel space)
- Η διαχείριση των **User-Level Threads** (δημιουργία, καταστροφή, etc) γίνεται μέσω βιβλιοθηκών στο user-space (επομένως είναι αποδοτική - δεν γίνονται system calls!)



19-11

Νήματα Επιπέδου Πυρήνα

- Τα σύγχρονα λειτουργικά συστήματα προσφέρουν τη δυνατότητα δημιουργίας νημάτων επιπέδου πυρήνα (kernel-level threads)
- Ο πυρήνας διαχωρίζει την έννοια της διεργασίας από τα νήματα
  - Διεργασία: οι πόροι που χρειάζεται ένα πρόγραμμα για να εκτελεστεί
  - Νήματα: κατάσταση εκτέλεσης της διεργασίας
- Ένα πρόγραμμα αποτελείται από μια διεργασία και από ένα ή περισσότερα νήματα επιπέδου πυρήνα
- Το λειτουργικό σύστημα προσφέρει μια επιπλέον κλήση συστήματος για τη δημιουργία ενός νέου νήματος επιπέδου πυρήνα στα πλαίσια μιας διεργασίας
- Τα νήματα επιπέδου πυρήνα είναι οντότητες αναγνωρίσιμες από το λειτουργικό σύστημα
- Ο χρονοπρογραμματιστής του λειτουργικού αναθέτει τα νήματα στους φυσικούς επεξεργαστές για να εκτελεστούν

Νήματα Επιπέδου Πυρήνα

- Η δημιουργία, καταστροφή και διαχείριση των νημάτων επιπέδου πυρήνα απαιτεί τη χρήση κατάλληλων κλήσεων συστήματος
- Υψηλός κόστος λόγω της μετάβασης του επεξεργαστή από το επίπεδο χρήστη στο επίπεδο πυρήνα
- Κάποιες εφαρμογές απαιτούν χρήση μεγάλου αριθμού νημάτων
  - Εκμετάλλευση λεπτά κατανεμημένου παραλληλισμού (fine-grained parallelism)
  - Εκτέλεση εμφωλευμένων (nested) βρόχων – παραλληλισμός
  - Καλύτερη εξισορρόπηση φορτίου
  - Αξιοπίστη αναδρομικών αλγορίθμων
  - Επικάλυψη υπολογισμών με επικοινωνία
- Το κόστος δημιουργίας – διαχείρισης των νημάτων επιπέδου πυρήνα υπερκαλύπτει το όφελος χρήσης τους

Νήματα Επιπέδου Χρήστη

- Εναλλακτική προσέγγιση: δημιουργία νημάτων στο επίπεδο χρήστη
- Τα νήματα αποτελούνται από ένα περιγραφέα (descriptor) που περιέχει
  - Τις τιμές των καταχωρητών του νήματος (registers)
  - Μία περιοχή μνήμης που χρησιμοποιείται ως στοίβα (stack)
  - Επιπλέον πεδία που χρησιμοποιούνται από τη βιβλιοθήκη νημάτων για διαχειριστικούς λόγους
- Τα νήματα επιπέδου χρήστη είναι οντότητες μη αναγνωρίσιμες από το λειτουργικό σύστημα
- Δημιουργούνται, διαχειρίζονται, χρονοδρομολογούνται και καταστρέφονται αποκλειστικά από κατάλληλη βιβλιοθήκη επιπέδου χρήστη

Νήματα Επιπέδου Χρήστη

- Για κάθε εργασία που μπορεί να εκτελεστεί παράλληλα δημιουργείται ένα νήμα επιπέδου χρήστη
- Το νήμα αυτό εκτελείται από κάποιο νήμα επιπέδου πυρήνα
- Τα νήματα επιπέδου πυρήνα λειτουργούν ως ιδεατοί επεξεργαστές (virtual processors) οι οποίοι εκτελούν νήματα επιπέδου χρήστη
  - Αντίστοιχα οι φυσικοί επεξεργαστές εκτελούν νήματα επιπέδου πυρήνα
  - Επομένως η δημιουργία και εκτέλεση πολλών νημάτων επιπέδου χρήστη με ένα νήμα επιπέδου πυρήνα δεν σημαίνει παράλληλη εκτέλεση
  - Τα νήματα επιπέδου χρήστη θα εκτελεστούν σε έναν φυσικό επεξεργαστή

8. Τι είναι η διεργασία, τι είναι το νήμα και ποιες οι διαφορές τους;

Απάντηση

Διεργασία είναι μια εφαρμογή που εκτελείται και διαχειρίζεται από το λειτουργικό σύστημα

✓ Είναι Ανεξάρτητη μονάδα εκτέλεσης και περιλαμβάνει:
- Πληροφορίες κατάστασης (State information)
- Ιδεατό χώρο μνήμης (Virtual memory)

✓ Η Επικοινωνία μεταξύ διεργασιών με σαφώς προκαθορισμένους τρόπους

γίνεται

Νήμα είναι ένα μονοπάτι εκτέλεσης εντολών εντός μιας διεργασίας

✓ Κάθε διεργασία έχει τουλάχιστον ένα νήμα

✓ Ολα τα νήματα μοιράζονται τους πόρους της διεργασίας
- Πολύ σημαντικός πόρος είναι ο χώρος διευθύνσεων ιδεατής μνήμης ✓

- Όλα τα νήματα έχουν άμεση επικοινωνία μεταξύ τους και έχουν άμεση πρόσβαση σε όλες τις μεταβλητές
- Το νήμα μπορεί να υποστεί διαχείριση:
  - ✓ είτε από το λειτουργικό σύστημα (Kernel-level thread)
  - ✓ είτε στο χώρο διευθύνσεων μνήμης του χρήστη (User-level thread). Αυτό γίνεται συνήθως σε μια βιβλιοθήκη νημάτων

Επιπλέον Διαφορές Διεργασίας και Νήματος είναι οι ακόλουθες:

Τα νήματα έχουν πολλές ομοιότητες με τις διεργασίες αλλά και δύο διαφορές:

- Έχουν όλα πρόσβαση σε μια κοινή περιοχή της μνήμης
- Η διαδικασία εναλλαγής τους στην ΚΜΕ είναι απλούστερη από την αντίστοιχη διαδικασία εναλλαγής διαδικασιών. CPU

Για να αναπαραστήσουμε ένα νήμα χρειαζόμαστε απαραίτητα 2 στοιχεία: 2 (ερώτηση)

- Program Counter
- Stack

9. Πως γίνεται η αντιμετώπιση διακοπής από τον επεξεργαστή: (όλη)

Απάντηση

Αλγόριθμος για **μεταχείριση interrupt**:
1. **Hardware:** (στο hardware)

1. Τοποθέτησε (push) στο stack τις τιμές των PC, SP, PSW, και κάποιων CPU registers
2. Βρες την κατάλληλη διεύθυνση στο interupt vector και "φόρτωσε" την στο PC. ⇒ Έτσι, εκτελείται η ISR.

Αλγόριθμος για **μεταχείριση interrupt**:
2. **Software:** (στο software)

1. Η ISR σώζει/αποθηκεύει τα **CPU registers** στο κατάλληλο struct του Process Table (assembly lang).
2. Μετά, το **SP** ενημερώνεται να "δείχνει" σ' ένα άλλο προσωρινό stack (assembly lang)
3. Μετά, καλείται μια C ρουτίνα που βρίσκει σε **ποιο process** αφορά αυτό το interrupt. Αυτό το process τώρα γίνεται "έτοιμο" (δηλ. το process struct φεύγει από την Sleep λίστα και πηγαίνει στην Ready λίστα).
4. Καλείται ο **scheduler** να διαλέξει ένα Process
5. Η C ρουτίνα επιστρέφει στην assembly lang. ρουτίνα που φορτώνει τα CPU Register και Page Tables του process που διάλεξε ο scheduler.

10.Περιγράψτε τις βασικές ιδέες πίσω από τη λειτουργία του μηχανισμού DMA και Block Interleaving.

Απάντηση

- DMA ονομάζεται η ικανότητα των ελεγκτών να εκτελέσουν πλήρως μια εντολή I/O χωρίς τη μεσολάβηση της CPU. Για παράδειγμα μπορεί ένας ελεγκτής να ανακτήσει το ζητούμενο μπλοκ από το δίσκο και να το μεταφέρει στην κύρια μνήμη χωρίς την ανάγκη να επέμβει η CPU και να κάνει την αντιγραφή από το buffer ελεγκτή στην κύρια μνήμη. Η ικανότητα DMA ενός ελεγκτή είναι πολύ σημαντική γιατί όσο χρόνο διαρκεί το DMA, η CPU εκτελεί άλλες εντολές και έτσι αυξάνεται ο βαθμός παραλληλισμού του συστήματος διότι εκτελούνται ταυτόχρονα εντολές και από τη CPU και από τον ελεγκτή και βελτιώνεται η απόδοση του συστήματος.

- Ένα βασικό πρόβλημα που προκύπτει από την αντικανότητα μερικών ελεγκτών για είσοδο και έξοδο ταυτόχρονα είναι όταν αρχίζουν DMA για ένα μπλοκ στον αποταμιευτή (buffer) τους ? το επόμενο ζητούμενο μπλοκ περνά κάτω από την κεφαλή και δεν μπορεί να αποθηκευτεί σε ένα άλλο αποταμιευτή. Μόλις τελειώσει το 1° DMA θα πρέπει να περιμένει ο ελεγκτής μέχρι το επόμενο ζητούμενο μπλοκ να ξαναπεράσει κάτω από την κεφαλή και αυτό είναι μια σημαντική καθυστέρηση. Μια λύση στο πρόβλημα αυτό βασίζεται στην έννοια του block interleaving. Αντί τα μπλοκ σε μια τροχιά του δίσκου να αποθηκεύονται με τη σειρά 1, 2, 3... αποθηκεύονται με τη σειρά 1, 5, 2, 6, 3, 7, 4, 8 κ.λ.π. Έτσι αντί για την καθυστέρηση μιας σχεδόν ολόκληρης

Τι είναι DMA και σε τι χρησιμεύει;

περιστροφής για να ανακτηθεί το block #2 (όσο το μπλοκ #1 γίνεται DMA) υπάρχει μόνο η καθυστέρηση να περάσει το block #5 κάτω από την κεφαλή.

11.Περιγράψτε τις διαφορές και δώστε ένα ορισμό για τις preemptive και non-preemptive πολιτικές δρομολόγησης

Απάντηση

Οι αλγόριθμοι χρονοπρογραμματισμού μπορούν να διαιρεθούν σε 2 κατηγορίες σε σχέση με τον τρόπο που αντιμετωπίζουν τις διακοπές ρολογιού. Οι μη προεκτοπιστικοί (non-preemptive) αλγόριθμοι χρονοπρογραμματισμού επιλέγουν μια διεργασία και της επιτρέπουν να εκτελεστεί μέχρι να μπλοκαριστεί (από I/O ή γιατί περιμένει κάποια άλλη διεργασία) ή μέχρι να επιστρέψει εθελοντικά τον έλεγχο της CPU. Συνεπώς στην περίπτωση αυτή δεν λαμβάνονται αποφάσεις χρονοπρογραμματισμού κατά τη διάρκεια των διακοπώ ρολογιού.

Αντίθετα οι προεκτοπιστικοί (non-preemptive) αλγόριθμοι χρονοπρογραμματισμού επιλέγουν κάθε φορά τη διεργασία που θα εκτελεστεί στη συνέχεια αλλά δεν της επιτρέπουν να εκτελεστεί περισσότερο από ένα συγκεκριμένο χρονικό διάστημα. Αν η εκτελούμενη διεργασία συνεχίσει να εκτελείται και μετά το πέρας του διαστήματος αυτού, το σύστημα την αναστέλλει υποχρεωτικά και ο χρονοπρογραμματιστής επιλέγει κάποια άλλη διεργασία (αν υπάρχει έστω και μια διαθέσιμη). Για να υλοποιηθεί ο προεκτοπιστικός χρονοπρογραμματισμός απαιτείται να ρυθμιστεί το ρολόι ώστε να προκαλεί διακοπές ακριβώς τη στιγμή που τελειώνει κάθε φορά το προκαθορισμένο χρονικό διάστημα ώστε να επιστρέφει με αυτόν τον τρόπο ο έλεγχος της στο χρονοπρογραμματιστή.

12.Ποια είδη αδιεξόδων γνωρίζετε; Ποια η διαφορά του αδιέξοδου από τη λιμοκτονία;

Απάντηση

➢ 1 Ένα σύνολο διεργασιών βρίσκεται σε (κανονικό) αδιέξοδο αν κάθε διεργασία του συνόλου περιμένει ένα συμβάν που μόνο μια άλλη διεργασία του συνόλου μπορεί να προκαλέσει

➢ 2 Ένα διαφορετικό είδος αδιεξόδου μπορεί να εμφανιστεί σε συστήματα επικοινωνίας όπου 2 ή περισσότερες διεργασίες επικοινωνούν μεταξύ τους **ανταλλάσσοντας μηνύματα**. Μια συνήθης μορφή είναι όταν η διεργασία Α στέλνει ένα μήνυμα στη διεργασία Β και μετά μπλοκάρει μέχρι η Β να της στείλει ένα μήνυμα απάντησης. Η Β θα παραμείνει μπλοκαρισμένη μέχρι να λάβει ένα αίτημα που θα της ζητάει να κάνει κάτι. Έτσι έχουμε ένα αδιέξοδο μόνο που αυτό δεν είναι το κλασσικό αδιέξοδο πόρων. Η Α δεν κατέχει κανένα πόρο που χρειάζεται η Β ούτε αντίστροφα. Για την ακρίβεια δεν υπάρχουν καθόλου πόροι στο σύστημα. Είναι όμως αδιέξοδο αφού έχουμε μια ομάδα διεργασιών που είναι όλες μπλοκαρισμένες περιμένοντας για ένα συμβάν το οποίο μπορεί να προκαλέσει η άλλη. Η κατάσταση αυτή ονομάζεται **αδιέξοδο επικοινωνίας**.

➢ 3 Μια άλλη περίπτωση αδιεξόδου είναι σε περιπτώσεις με πολυάσχολη αναμονή (busy wait) για την είσοδο σε μια κρίσιμη περιοχή ή για την προσπέλαση ενός πόρου. Έστω ότι έχουμε ένα ζεύγος διεργασιών που ζητούν πόρους όπως ακολούθως:

void process_A()	void process_B()
{	{
enter_region(resource1);	enter_region(resource2);
enter_region(resource2);	enter_region(resource1);
use_both_resources();	use_both_resources();
leave_region(resource2);	leave_region(resource1);
leave_region(resource1);	leave_region(resource2);
}	}

Κάθε διεργασία χρειάζεται 2 πόρους και χρησιμοποιεί την εντολή enter_region για να αποκτήσει τα απαραίτητα κλειδώματα. Αν η προσπάθεια αποτύχει τότε η διεργασία προσπαθεί πάλι. Αν εκτελεστεί πρώτα η διεργασία Α και αποκτήσει την πόρο resource1 και μετά εκτελεστεί η διεργασία Β και αποκτήσει τον πόρο resource2, όποια διεργασία και να εκτελεστεί στη συνέχεια δεν πρόκειται να κάνει καμία πρόοδο αλλά και καμία δεν θα μπλοκάρει. Απλώς κάθε μια θα χρησιμοποιεί το κβάντο χρόνου της CPU που της έχει εκχωρηθεί ξανά και ξανά χωρίς να κάνει κάτι ωφέλιμο αλλά και χωρίς να μπλοκάρει. Έτσι δεν έχουμε αδιέξοδο (αφού δεν υπάρχει μπλοκαρισμένη διεργασία) αλλά έχουμε κάτι που είναι λειτουργικά ισοδύναμο με το αδιέξοδο και ονομάζεται **ενεργό αδιέξοδο** (livelock)

➢ **Λιμοκτονία** όταν μια διεργασία περιμένει στην ουρά αναμονής να λάβει τη CPU για να εκτελεστεί και εισέρχονται στην ουρά νέες διεργασίες με μεγαλύτερη προτεραιότητα οπότε αναβάλλεται συνεχώς η εκτέλεση της τρέχουσας διεργασίας. Σε αντίθεση με το αδιέξοδο και το ενεργό αδιέξοδο η λιμοκτονία μπορεί να σταματήσει από μόνη της όταν τελειώσει η εκτέλεση όλων των υπαρχόντων διεργασιών υψηλότερης προτεραιότητας και δεν φτάσουν νέες διεργασίες υψηλότερης προτεραιότητας.

13.Ποιοι οι τρόποι προσπέλασης ενός αρχείου;

Απάντηση

α)Σειριακή (Sequential Access:): Όλα τα bytes ενός αρχείου προσπελαύνονται κατά σειρά από το 1ο μέχρι το τελευταίο μπλοκ δηλαδή για να προσπελάσουμε το n-οστό μπλοκ πρέπει να προσπελάσουμε πρώτα όλα τα προηγούμενα. Αυτό ο τρόπος προσπέλασης εφαρμόζεται σε μαγνητικές ταινίες.

β)Τυχαία (Random Access): Όταν τα bytes/records ενός αρχείου προσπελαύνονται κατά μια τυχαία σειρά. Αυτό ο τρόπος προσπέλασης εφαρμόζεται στο σκληρό δίσκο. Η τυχαία προσπέλαση υλοποιείται με δύο τρόπους: 1)κάθε read() & write() προσδιορίζει που ακριβώς στο αρχείο θα επιδράσει η λειτουργία δηλ. ποιο θα είναι το 1° byte. Για παράδειγμα στην εντολή read(fd,&buf,100) διαβάζονται 100 bytes από το αρχείο που δείχνει ο δείκτης fd και τοποθετούνται στον πίνακα buf και μπορεί η εντολή να κρατά ένα δείκτη στο 1° byte του αρχείου. β)Το σύστημα αρχείων (File System-F.S.) κρατάει ένα read/write δείκτη (για κάθε process που προσπελαύνει ένα αρχείο). Το 1° byte είναι αυτό στο οποίο δείχνει ο δείκτης. Υπάρχει και ένα system call seek() με το οποίο ο χρήστης μπορεί να ενημερώσει το δικό του r/w pointer

14.Ποιο το πλεονέκτημα του system call Map(filename, virtual-address) και ποια τα πλεονεκτήματα και τα μειονεκτήματα των αρχείων που είναι χαρτογραφημένα στη μνήμη;

Απάντηση

- Το system call Map(filename, virtual-address) φορτώνει όλα τα αρχεία μιας διεργασίας στην κύρια μνήμη, μαζί με τη διεργασία, οπότε οποιαδήποτε αναφορά στα αρχεία αυτά αποφεύγει την πρόσβαση στο δίσκο.
- Τα χαρτογραφημένα αρχεία παρέχουν ένα εναλλακτικό μοντέλο εισόδου/εξόδου. Αντί για την πραγματοποίηση αναγνώσεων και εγγραφών είναι δυνατή η πρόσβαση στο αρχείο με τη μορφή ενός μεγάλου πίνακα χαρακτήρων στη μνήμη. Αν δύο ή περισσότερες διεργασίες χαρτογραφούν στο ίδιο αρχείο την ίδια στιγμή, μπορούν να επικοινωνούν μέσω της κοινόχρηστης μνήμης. Οι εγγραφές που κάνει μια διεργασία στην κοινόχρηστη μνήμη είναι αμέσως ορατές όταν η άλλη διαβάζει από το τμήμα του χώρου εικονικών διευθύνσεων της που έχει χαρτογραφηθεί στο αρχείο.
- Τα αρχεία που είναι χαρτογραφημένα στη μνήμη είναι αυτά που έχουν φορτωθεί στην κύρια μνήμη μαζί με τη διεργασία που τα χρησιμοποιεί. Τα page faults εξυπηρετούνται προσπελαύνοντας τα κατάλληλα disk blocks (από το swap space). Επίσης το process προσπελαύνει ένα αρχείο όπως οποιαδήποτε μεταβλητή/δομή δεδομένων του προγράμματός του.
- Το βασικό μειονέκτημα των αρχείων που είναι χαρτογραφημένα στην μνήμη είναι εκτός από τα προβλήματα ασυνέπειας υπάρχουν και προβλήματα διαμοιρασμού καθώς και προβλήματα του να χωράει αρχείο στην κύρια μνήμη. Πιο συγκεκριμένα είναι δύσκολη η ταυτόχρονη χρήση από δύο διεργασίες που το χρειάζονται και το ενημερώνουν ταυτόχρονα και επίσης τα αρχεία λόγω του μεγέθους τους δεν χωράνε στην κύρια μνήμη.

15.Ποιές οι βασικές στρατηγικές ανάθεσης block σε αρχεία

Απάντηση

α)Συνεχόμενη Ανάθεση

Τα πλεονεκτήματα της συνεχόμενης ανάθεσης είναι

1) Όταν ένα αρχείο χρειάζεται N disk blocks τότε N συνεχόμενα blocks ανατίθενται στο αρχείο αυτό
2) Αυτή η στρατηγική είναι απλή και έχει το πλεονέκτημα ότι για να διαβαστεί ολόκληρο το αρχείο απαιτείται να πληρωθεί το κόστος για το disk seek (αναζήτηση στο δίσκο για τα blocks του αρχείου) και η καθυστέρηση περιστροφής δίσκου (disk rotational delay) μόνο μια φορά. Αυτό είναι σημαντικό εξ αιτίας των σχετικών μεγάλων κοστών

Τα μειονεκτήματα της συνεχόμενης ανάθεσης είναι:

- συχνά δεν είναι γνωστό από πριν (και είναι μεταβαλλόμενο) το μέγιστο μέγεθος του αρχείου οπότε το σύστημα αρχείων δεν μπορεί να ξέρει πόσα disk blocks να αναθέσει.
- όταν τα αρχεία διαγράφονται δημιουργούνται προβλήματα κατακερματισμού. Πιο συγκεκριμένα ένας δίσκος μπορεί να διαθέτει συνολικά N ελεύθερα blocks και ένα αρχείο που χρειάζεται N blocks να μην μπορεί να δημιουργηθεί λόγω του ότι τα N ελεύθερα blocks του δίσκου δεν είναι συνεχόμενα.

β)Ανάθεση με Συνδεδεμένες Λίστες

Τα πλεονεκτήματα της ανάθεσης με συνδεδεμένες λίστες είναι:

Με αυτή τη στρατηγική τα αρχεία είναι μια λίστα από blocks με μερικά bytes του κάθε block να δείχνουν στο επόμενο block της λίστας και έτσι αποφεύγεται το πρόβλημα κατακερματισμού. Επίσης κάθε εγγραφή καταλόγου αποθηκεύει μόνο τη διεύθυνση του 1ου block του αρχείου.

Τα μειονεκτήματα της ανάθεσης με συνδεδεμένες λίστες είναι:

Το βασικό πρόβλημα αυτής της μεθόδου είναι ότι όταν επιθυμείται τυχαία προσπέλαση η απόδοση του συστήματος είναι πολύ χαμηλή διότι όταν επιθυμούμε να προσπελάσουμε το n-οστό block ενός αρχείου θα πρέπει να προσπελαστούν απαραίτητα τα n-1 προηγούμενα block. Το πρόβλημα αυτό μπορεί να λυθεί χρησιμοποιώντας ένα χάρτη δεικτοδότησης (index) στην κύρια μνήμη που περιέχει τους δείκτες για τα disk blocks

γ)Δεικτοδοτημένη ανάθεση (indexed allocation)

Τα πλεονεκτήματα της δεικτοδοτημένης ανάθεσης είναι:

Με καθολικό πίνακα (πίνακας στην κύρια μνήμη που περιέχει όλα τα blocks όλων των αρχείων του σκληρού δίσκου) ο πίνακας βρίσκεται πάντα στην κύρια μνήμη και τα παρασπάνισια disk I/O ελαχιστοποιούνται.

Τα μειονεκτήματα της δεικτοδοτημένης ανάθεσης είναι:

Ο καθολικός πίνακας καταναλώνει πολύ κύρια μνήμη γιατί διαθέτει μια εγγραφή για κάθε block του δίσκου και έτσι δεν μπορεί να αποθηκεύει όλα τα blocks όλων των αρχείων του δίσκου .

16.Να περιγράψετε τη διαδικασία μετάφρασης ονόματος όταν δέχεται σαν είσοδο το μονοπάτι /d1/d2/d3/foo.
Απάντηση
Α.
α)Το inode για το root directory "/" ανακτάται από τον δίσκο. (Αυτό το inode βρίσκεται σε μια γνωστή και μη ενημερώσιμη διεύθυνση στο δίσκο).
β)Από το inode του root βρίσκονται τα disk blocks (που περιέχουν τα dir. entries), ανακτώνται και ψάχνονται για την εγγραφή "d1". Όταν βρίσκεται η εγγραφή για το "d1" τότε βρίσκεται και το inode# του d1.
Β.
α)Το inode για το "d1" ανακτάται απ' το δίσκο.
β)Τα disk blocks του dir file "d1" ανακτώνται και ψάχνονται για την εγγραφή "d2"....
γ)Τέλος το inode & disk blocks του "dir3" ανακτώνται και ψάχνονται για την εγγραφή "foo".
δ)Έτσι βρίσκεται το inode# για το αρχείο «foo» το οποίο ανακτάται και τοποθετείται στη μνήμη (σε ένα cache ειδικό για inodes που λέγεται inode table).

Γ. Αφού τοποθετηθεί το inode στο inode table του πυρήνα ο file descriptor πίνακας του process ενημερώνεται.
Δ. Μια εγγραφή του χρησιμοποιείται για έναν δείκτη ακολουθώντας τον οποίο μπορεί ο kernel να βρει το inode στο inode table.
Ε. Τέλος επιστρέφεται ο δείκτης της εγγραφής του file descriptor πίνακα...

17.Να περιγράψετε τα πλεονεκτήματα χρήσης μνήμης cache τόσο στο διαχειριστή μνήμης όσο και στο διαχειριστή αρχείων
Απάντηση
Διαχειριστής Μνήμης
Αφού οι πίνακες σελίδων είναι πολύ μεγάλοι για να χωρέσουν σε CPU registers μπορούμε να μειώσουμε το κόστος πρόσβασης σε' αυτούς στη κύρια μνήμη με τη χρήση συσχετιστικής μνήμης (associative memory) που λειτουργεί σαν «κρυφή μνήμη» ή «προσωρινή μνήμη» (cache). Η θεμελιώδης παρατήρηση είναι ότι η συμπεριφορά των περισσότερων προγραμμάτων είναι τέτοια ώστε να παρατηρούνται μεγάλοι αριθμοί αναφορών σε λίγες σελίδες (που αλλάζουν αργά με το πέρασμα του χρόνου) π.χ. - όλες οι εντολές σε μια σελίδα εντολών (βλέπε loops). Έτσι χρησιμοποιείται ειδικό υλικό που αποτελείται από καταχωρητές το σύνολο των οποίων καλούνται TLB -- translation lookaside buffer. Ο αριθμός αυτών των registers είναι μικρός (π.χ. ≤ 64).

Διαχειριστής Αρχείων
Η επίτευξη καλύτερης απόδοσης στο **διαχειριστή Αρχείων** βασίζεται στην έννοια της F.S. buffer cache. Η κρυφή μνήμη είναι ένα τμήμα της K.M. (στον πυρήνα) το οποίο αποθηκεύει προσωρινά μερικά μπλοκ του δίσκου. Όταν κάποια διεργασία ζητεί πρόσβαση σε κάποιο μπλοκ, ο πυρήνας
- φορτώνει πρώτα αυτό το μπλοκ στην κρυφή μνήμη και
- μετά το δίνει στη διεργασία.

Οι μεταγενέστερες αιτήσεις (από την ίδια ή άλλες διεργασίες) για το ίδιο μπλοκ θα εξυπηρετηθούν από τη κρυφή μνήμη και έτσι αποφεύγεται το disk I/O. Έτσι, κάθε φορά που ζητείται κάποιο μπλοκ, ο kernel πρώτα ψάχνει τη κρυφή μνήμη και μόνο αν δεν το βρει απευθύνεται στο δίσκο.

18.Να αναφέρετε ένα πλεονέκτημα των πραγματικών συνδέσμων σε σχέση με τους συμβολικούς και ένα πλεονέκτημα των συμβολικών συνδέσμων σε σχέση με τους πραγματικούς.
Απάντηση
Ο σύνδεσμος (link) είναι μια τεχνική που επιτρέπει σε ένα αρχείο να εμφανίζεται σε περισσότερους από ένα καταλόγους. Η κλήση συστήματος (link) στο Unix καθορίζει ένα υπάρχον αρχείο και ένα όνομα διαδρομής και δημιουργεί ένα σύνδεσμο από το υπάρχουν αρχείο προς το όνομα που καθορίζει η διαδρομή. Με τον τρόπο αυτό το ίδιο αρχείο μπορεί να εμφανίζεται σε περισσότερους καταλόγους. Οι σύνδεσμοι αυτού του είδους αυξάνουν το μετρητή του i-node του αρχείου για να παρακολουθούν τον αριθμό των καταλόγων που περιέχουν το αρχείο και ονομάζονται **πραγματικοί σύνδεσμοι (hard links).**
Μια παραλλαγή της ιδέας των συνδέσμων είναι ο **συμβολικός σύνδεσμος** (symbolic link). Αντί να έχουμε δύο ονόματα που δείχνουν στην ίδια εσωτερική δομή δεδομένων που αναπαριστά ένα αρχείο, να μπορούμε να δημιουργήσουμε ένα όνομα το οποίο δείχνει σε ένα μικροσκοπικό αρχείο που κατονομάζει ένα άλλο αρχείο. Όταν χρησιμοποιείται το πρώτο αρχείο το σύστημα αρχείων ακολουθεί τη διαδρομή και βρίσκει το όνομα στο τέλος της. Στη συνέχεια από την αρχή τη διαδικασία αναζήτησης χρησιμοποιώντας το νέο όνομα. Οι συμβολικοί σύνδεσμοι έχουν το πλεονέκτημα ότι μπορούν να ξεπερνούν τα όρια ενός δίσκου και ακόμη και να δίνουν ονόματα σε αρχεία απομακρυσμένων υπολογιστών. Η υλοποίηση τους είναι ελαφρώς λιγότερο αποδοτική από τους πραγματικούς συνδέσμους.

19. Έχει προταθεί το πρώτο τμήμα κάθε αρχείου στο Unix να διατηρείται στο ίδιο μπλοκ δίσκου με το αντίστοιχο i-node του αρχείου. Ποιο το πλεονέκτημα αυτής ης προσέγγισης;
Απάντηση
Πολλά αρχεία στο Unix είναι μικρά. Αν ολόκληρο το αρχείο χωρά στο ίδιο block με το i-node του αρχείου, τότε θα χρειαστεί μόνο μια πρόσβαση στο δίσκο για το διάβασμα του αρχείου, αντί για 2 προσβάσεις όπως είναι η παρούσα κατάσταση. Ακόμα και για με-

γαλύτερα αρχεία η διατήρηση του πρώτου τμήματος του αρχείου στο ίδιο block δίσκου με το i-node του αρχείου θα είχε κέρδος γιατί θα απαιτούνταν γενικά λιγότερες προσβάσεις στο δίσκο για το διάβασμα του αρχείου.

20. Ποιους τύπους αρχείων γνωρίζετε;

Απάντηση

- **Τα κανονικά αρχεία** είναι αυτά που περιέχουν πληροφορίες των χρηστών. Τα κανονικά αρχεία μπορεί να είναι είτε αρχεία ASCII που αποτελούνται από γραμμές κειμένου είτε δυαδικά αρχεία (binary)
- **Τα αρχεία καταλόγων** είναι αρχεία συστήματος που βοηθούν στην οργάνωση του συστήματος αρχείων
- **Τα ειδικά αρχεία χαρακτήρων** σχετίζονται με την είσοδο/έξοδο και χρησιμοποιούνται για να μοντελοποιήσουν σειριακές συσκευές I/O όπως τα τερματικά, οι εκτυπωτές και τα δίκτυα.
- **Τα ειδικά αρχεία μπλοκ** χρησιμοποιούνται για να μοντελοποιήσουν δίσκους

21. Η απόδοση μπορεί να βελτιωθεί και να εξοικονομηθεί χώρος στο δίσκο αν τα δεδομένα των μικρών αρχείων αποθηκεύονται μέσα στον κόμβο i. Πόσα bytes δεδομένων μπορούν να αποθηκευτούν μέσα σε αυτόν;

Απάντηση

Θα πρέπει ένας τρόπος που να σηματοδοτεί αν ένα μπλοκ περιέχει δεδομένα ή δείκτες. Θα μπορούσαμε για το σκοπό αυτό να χρησιμοποιήσουμε ένα bit από τα attributes. Από τους 10 άμεσους δείκτες όπου ο καθένας έχει μέγεθος k bytes, θα μπορούσαμε να αποθηκεύσουμε στους πρώτους 9 δείκτες ένα αρχείο μεγέθους 9k bytes και ο $10^{ος}$ δείκτης να δείχνει σε block. Αν δεν χρησιμοποιηθεί κανένα bit από τα attributes για να σηματοδοτεί αν ένα μπλοκ περιέχει δεδομένα ή δείκτες τότε η $1^{η}$ διεύθυνση θα χρησιμοποιηθεί για να δείχνει αν ένα block περιέχει δεδομένα ή δείκτες και το αρχείο θα έχει μέγεθος 8k bytes

22. Πως εξυπηρετεί ο οδηγός τις αιτήσεις που λαμβάνει (αναλυτική επεξήγηση);

Απάντηση

- Ο οδηγός έχει μια ουρά εργασίας όπου μπαίνουν αιτήσεις για E/E (I/O). Η σειρά με την οποία αποθηκεύονται οι αιτήσεις στην ουρά ενός οδηγού είναι σημαντική για την απόδοση του συστήματος γιατί καθορίζει τη σειρά με την οποία θα ανακτηθούν τα ζητούμενα μπλοκ
- Ο οδηγός εκδίδει εντολή στον ελεγκτή και μετά μπλοκάρει. Θα τον ξεμπλοκάρει ο χειριστής διακοπών που θα καλέσει ο ελεγκτής όταν ολοκληρώσει την εντολή E/E
- Μετά θα εξετάσει τον κατάλληλο CSR register για τυχόν λάθη. Αν έχει υπάρξει λάθος, θα προσπαθήσει να το αντιμετωπίσει, αλλιώς, θα επικοινωνήσει με το αμέσως ανώτερο επίπεδο (δηλ. με το device independent software) για να του δώσει πληροφορία όπως π.χ. ένα disk block που ανακτήθηκε κ.λπ.
- Κατόπιν, αν η ουρά αναμονής του οδηγού δεν είναι άδεια, ο οδηγός θα αφαιρέσει την επόμενη αίτηση από αυτή και θα τη στείλει στον ελεγκτή. Αν η ουρά είναι άδεια, τότε ο οδηγός θα περιμένει την επόμενη αίτηση

23.Για ποιο λόγο χρησιμοποιούνται αποταμιευτές (buffers) στους ελεγκτές (controllers)

Απάντηση

Ο κύριος λόγος απορρέει από την ανάγκη χρησιμοποίησης του system bus για να μεταφερθεί πληροφορία από το δίσκο στην κύρια μνήμη. Το system bus χρησιμοποιείται από το CPU για πρόσβαση στη μνήμη και για επικοινωνία CPU-δίσκων. Έτσι τη στιγμή που τα bits ενός μπλοκ καταφθάνουν στον ελεγκτή, το system bus μπορεί να μην είναι διαθέσιμο (δηλ. να μεταφέρει κάποια άλλη πληροφορία)

24. Ποιες συσκευές είναι block και ποιες συσκευές είναι character. Αναφέρετε παραδείγματα από κάθε κατηγορία

Απάντηση

- Οι συσκευές Block είναι αυτές που αποθηκεύουν πληροφορία σε μονάδες σταθερού μεγέθους (# bytes) που λέγονται blocks. Κάθε block πληροφορίας έχει τη δική του διεύθυνση και μπορεί να εγγραφεί ή να ανακτηθεί ανεξάρτητα από τα άλλα blocks. Το block αποτελεί τη μικρότερη μονάδα πληροφορίας που μεταφέρεται από (προς) την συσκευή.
- Οι συσκευές Character διαχειρίζονται αδόμητη πληροφορία, δηλαδή απλώς μια σειρά από χαρακτήρες. Δεν υπάρχει η δυνατότητα διευθυνσιοποίησης πληροφορίας και ανεξάρτητης πρόσβασης.
- Οι δίσκοι και οι μαγνητικές ταινίες είναι block devices ενώ τα τερματικά, οι εκτυπωτές, τα δίκτυα, τα ποντίκια είναι character devices

25.Ποιες βελτιώσεις πρέπει να γίνουν στον ελεγκτή ενός δίσκου ώστε να αυξηθεί η απόδοση του;

Απάντηση

- Προανάκτηση τομέων δηλαδή μετά την αναζήτηση κυλίνδρου και την καθυστέρηση περιστροφής, δεν ανακτάται μόνο ο ζητούμενος τομέας αλλά και μερικοί επόμενοι τομείς (sectors) ή και όλοι οι τομείς της τροχιάς (track)
- Οι επιπλέον τομείς τοποθετούνται στην κρυφή μνήμη του ελεγκτή και ανακτώνται από εκεί αν ζητηθούν. Έτσι αποφεύγεται το κόστος πρόσβασης στη μαγνητική επιφάνεια του δίσκου

26.Ποια η διαφορά μεταξύ διαμόρφωσης χαμηλού επιπέδου και διαμόρφωσης υψηλού επιπέδου;

Απάντηση

Πριν χρησιμοποιηθεί ένας δίσκος πρέπει να περάσει από **διαμόρφωση χαμηλού επιπέδου** (low level format) μέσω λογισμικού. Η διαμόρφωση δημιουργεί μια σειρά ομόκεντρων τροχιών (κυλίνδρων-tracks) καθεμία από τις οποίες περιέχει ένα συγκεκριμένο αριθμό τομέων (sectors) που χωρίζονται με μικρά κενά. Η θέση του τομέα 0 σε κάθε τροχιά είναι η σχετική απόσταση από την προηγούμενη τροχιά και ονομάζεται **στρέβλωση κυλίνδρου** (cylinder skew) και χρησιμοποιείται για να βελτιωθεί η απόδοση. Η βασική ιδέα της μεθόδου αυτής είναι να επιτραπεί στο δίσκο να διαβάζει πολλές τροχιές με μια συνεχή λειτουργία χωρίς να χάνει δεδομένα. Το τελευταίο βήμα της ετοιμασίας ενός δίσκου είναι η **διαμόρφωση υψηλού επιπέδου** (high level format) για το κάθε διαμέρισμα ξεχωριστά. Η λειτουργία αυτή δημιουργεί ένα μπλοκ εκκίνησης, τη διαχείριση ελεύθερου χώρου στο δίσκο (λίστα ή χάρτης bit), το βασικό κατάλογο και ένα κενό σύστημα αρχείων. Τοποθετεί επίσης ένα κωδικό σε κάθε καταχώριση του πίνακα διαμερισμάτων ο οποίος αναφέρει το σύστημα αρχείων που χρησιμοποιείται στο συγκεκριμένο διαμέρισμα επειδή διάφορα λειτουργικά συστήματα υποστηρίζουν πολλά και ασύμβατα μεταξύ τους συστήματα αρχείων. Στο σημείο αυτό μπορεί να ξεκινήσει το σύστημα.

27. Περιγράψτε συνοπτικά τις βασικές ιδέες πίσω από τη λειτουργία του μηχανισμού DMA (Direct Memory Access). Περιγράψτε συνοπτικά τα βήματα που περιλαμβάνει μια διαδικασία ανάγνωσης ενός block πληροφορίας από το σκληρό δίσκο στην κύρια μνήμη με τη χρήση DMA λογικής.

Απάντηση

DMA ονομάζεται η ικανότητα των ελεγκτών να εκτελέσουν πλήρως μια εντολή I/O χωρίς τη μεσολάβηση της CPU. Για παράδειγμα μπορεί ένας ελεγκτής να ανακτήσει το ζητούμενο μπλοκ από το δίσκο και να το μεταφέρει στην κύρια μνήμη απευθείας χωρίς την ανάγκη να επέμβει η CPU για να κάνει την αντιγραφή από το buffer ελεγκτή στην κύρια μνήμη. Η ικανότητα DMA ενός ελεγκτή είναι πολύ σημαντική γιατί όσο χρόνο διαρκεί το DMA, η CPU εκτελεί άλλες εντολές και έτσι αυξάνεται ο βαθμός παραλληλισμού του συστήματος διότι εκτελούνται ταυτόχρονα εντολές και από τη CPU και από τον ελεγκτή και βελτιώνεται η απόδοση του συστήματος.

5 **Θέματα Φεβρουάριος 2014**

ΘΕΜΑ 1
(i) Καταγράψτε τις βασικές τεχνικές που χρησιμοποιούνται για διαχείριση των ελεύθερων μπλοκ σε ένα σύστημα διαχείρισης αρχείων, τα πλεονεκτήματα και τα μειονεκτήματά τους.
(ii) Εξηγείστε ποιες οι διαφορές διεργασιών και νημάτων.

Απάντηση
ΘΕΜΑ 1
(i)

Διαχείριση Ελεύθερου Χώρου στο Δίσκο

Συνήθως, χρησιμοποιούνται 2 μέθοδοι:

□ **Συνδεδεμένες λίστες (Linked List):**

Κάθε κόμβος της λίστας είναι ένα μπλοκ δίσκου.
Κάθε κόμβος περιέχει:

- και N-1 διευθύνσεις ελεύθερων (free) μπλοκ, όπου N είναι ο αριθμός των διευθύνσεων που χωρούν σ' ένα μπλοκ δίσκου.
- την διεύθυνση του επόμενου κόμβου της λίστας
- (π.χ. με μέγεθος μπλοκ = 2K και με διευθύνσεις 4 ψηφίων, N = 2048/4).

□ **Χάρτης ψηφίων (Bit Map):**

- Ένας πίνακας (bit map) με N εγγραφές απαιτείται για ένα δίσκο με N blocks.
- Τα ελεύθερα μπλοκ αντιπροσωπεύονται με την τιμή 1 στον πίνακα.

→ για κάθε μπλοκ κρατείται πληροφορία μεγέθους 1 bit και όχι μερικά bytes όπως στη μέθοδο linked list

→ (εκτός μερικών περιπτώσεων) η μέθοδος bit map είναι προτιμότερη, από πλευράς χώρου.

□ Επίσης, επειδή το bit map είναι μικρότερο
→ υπάρχει μεγαλύτερη πιθανότητα να χωράει στη Κ.Μ.

- Αν ναι, τότε η μέθοδος bit map προτιμάται
- Αν όχι, τότε μπορεί κάθε block/κόμβος της λίστας (linked list) περιέχει συνήθως πιο πολλά ελεύθερα blocks απ' ότι ένα μπλοκ του bit map
- → ίσως να είναι προτιμότερη η μέθοδος linked list.

(ii)

ΔΙΑΦΟΡΕΣ ΔΙΕΡΓΑΣΙΩΝ-ΝΗΜΑΤΩΝ

Διεργασία είναι μια εφαρμογή που εκτελείται και διαχειρίζεται από το λειτουργικό σύστημα

✓ Είναι Ανεξάρτητη μονάδα εκτέλεσης και περιλαμβάνει:

- Πληροφορίες κατάστασης (State information)
- Ιδεατό χώρο μνήμης (Virtual memory)

- ✓ Η Επικοινωνία μεταξύ διεργασιών με σαφώς προκαθορισμένους τρόπους

Νήμα είναι ένα μονοπάτι εκτέλεσης εντολών εντός μιας διεργασίας

- ✓ Κάθε διεργασία έχει τουλάχιστον ένα νήμα
- ✓ Όλα τα νήματα μοιράζονται τους πόρους της διεργασίας
  - Πολύ σημαντικός πόρος είναι ο χώρος διευθύνσεων ιδεατής μνήμης
  - Όλα τα νήματα έχουν άμεση επικοινωνία μεταξύ τους και έχουν άμεση πρόσβαση σε όλες τις μεταβλητές
  - Το νήμα μπορεί να υποστεί διαχείριση:
    - ✓ είτε από το λειτουργικό σύστημα (Kernel-level thread)
    - ✓ είτε στο χώρο διευθύνσεων μνήμης του χρήστη (User-level thread). Αυτό γίνεται συνήθως σε μια βιβλιοθήκη νημάτων

Επιπλέον Διαφορές Διεργασίας και Νήματος είναι οι ακόλουθες:

Τα νήματα έχουν πολλές ομοιότητες με τις διεργασίες αλλά και δύο διαφορές:

- Έχουν όλα πρόσβαση σε μια κοινή περιοχή της μνήμης
- Η διαδικασία εναλλαγής τους στην ΚΜΕ είναι απλούστερη από την αντίστοιχη διαδικασία εναλλαγής διαδικασιών.

ΘΕΜΑ 2
Σε ένα δίσκο με 200 κυλίνδρους (αριθμημένους από 1 έως 200) δίνεται η ακόλουθη ακολουθία αιτήσεων: 20, 12, 34, 49, 43, 88, 65, 99, 110. Έστω ότι ο χρόνος μετακίνησης είναι 1 msec ανά κύλινδρο. Ποιος είναι ο συνολικός χρόνος εξυπηρέτησης αιτήσεων για τους ακόλουθους αλγορίθμους:
(α) Εξυπηρέτηση με βάση τη σειρά άφιξης (FCFS).
(β) Εξυπηρέτηση του πλησιέστερου κυλίνδρου (SSF).
(γ) Αλγόριθμος του ανελκυστήρα (Elevator- SCAN).
(δ) Αλγόριθμος σάρωσης προς την ίδια κατεύθυνση (C-SCAN).

<u>Απάντηση</u>
<u>ΘΕΜΑ 2</u>
Θεωρούμε ότι η κεφαλή αρχικά είναι στον κύλινδρο 10. Σελίδα 42 Άσκηση 13

FCFS
Οι κύλινδροι είναι:
10→20→12→34→49→43→88→65→99→110
10 + 8+ 22 + 15 + 6 + 45 + 23 + 34 + 11 = 174 κύλινδροι • 1 msec = 174 msec

SSF
Οι κύλινδροι είναι:
10→12→20→34→43→49→65→88→99→110
2 + 8 + 14 + 9 + 6 +16 + 23 + 11 + 11 = 100 κύλινδροι • 1 msec =100 msec

SCAN (προς τα επάνω)
10→12→20→34→43→49→65→88→99→110
2 + 8 + 14+ 9 + 6 + 16 + 23+ 11 + 11 = 100 κύλινδροι • 1 msec = 100 msec

Παρατήρηση
Αν υπήρχαν αιτήσεις π.χ. και για τους κυλίνδρους 5 και 8 τότε η κεφαλή μετά τον κύλινδρο 110 θα έκανε ακόμα τις εξής μετακινήσεις: 110→200→8→5

SCAN (προς τα κάτω)
10→1→12→20→34→43→49→65→88→99→110
9 + 11 + 8 + 14+ 9 + 6 + 16 + 23+ 11 + 11 = 118 κύλινδροι • 1 msec = 118 msec

C-SCAN (προς τα επάνω)
10→12→20→34→43→49→65→88→99→110
2 + 8 + 14+ 9 + 6 + 16 + 23+ 11 + 11 = 100 κύλινδροι • 1 msec = 100 msec

Παρατήρηση

Αν υπήρχαν αιτήσεις π.χ. και για τους κυλίνδρους 5 και 8 τότε η κεφαλή μετά τον κύλινδρο 110 θα έκανε ακόμα τις εξής μετακινήσεις: 110→200→1→5→8

C-SCAN (προς τα κάτω)

10→1→200→110→99→88→65→49→43→34→20→12

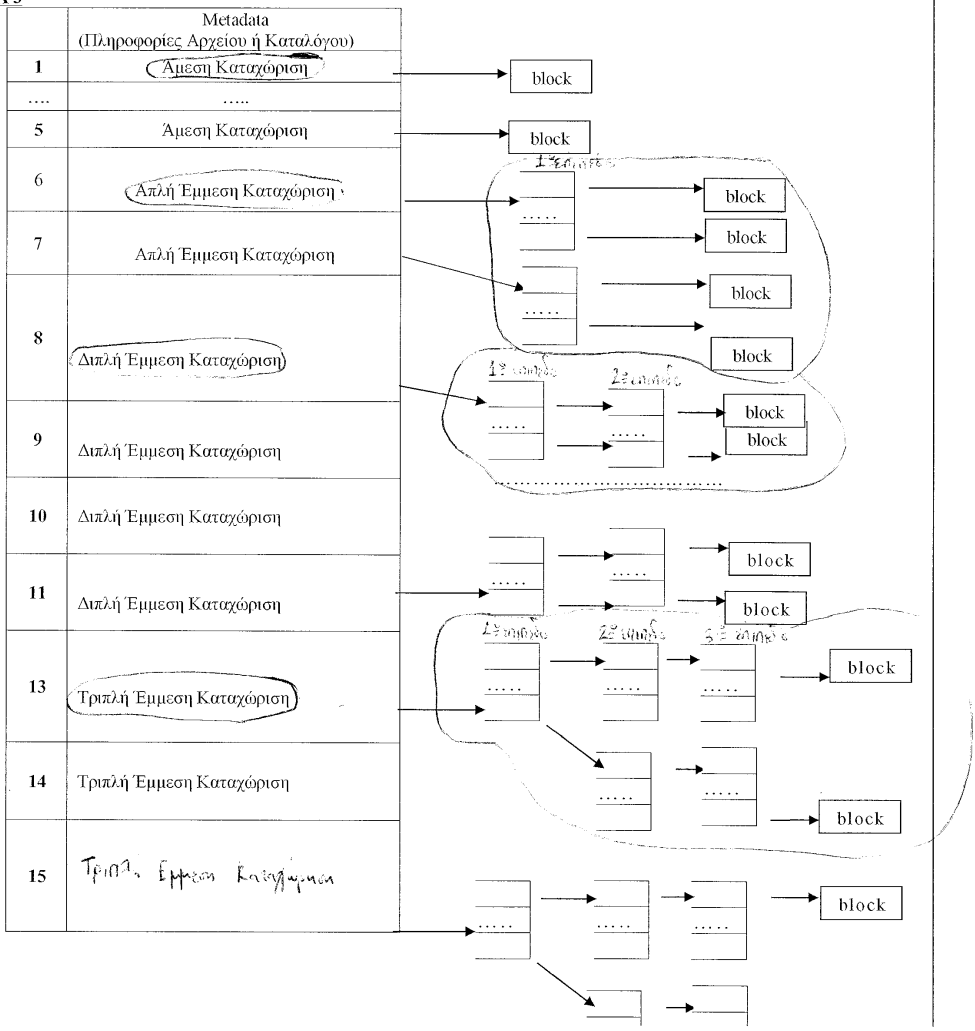
9 + 199+ 90 +11 + 11 + 23 + 16 + 6 + 9 + 14 + 8 = 396 κύλινδροι • 1 msec = 396 msec

ΘΕΜΑ 3

Στο σύστημα αρχείων UNIX δοκιμάζει μπλοκ με μέγεθος 8Κ Β και διευθύνσεις δίσκου με μέγεθος 16 bytes. Ποιο είναι το μέγιστο μέγεθος αρχείου αν οι κόμβοι i περιέχουν 5 άμεσες καταχωρήσεις, 2 απλές, 1 διπλές και 1 τριπλή έμμεση καταχώρηση;

Απάντηση

ΘΕΜΑ 3



- Ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από 5 άμεσες καταχωρήσεις δείκτες είναι:
  5 άμεσες καταχωρήσεις • μέγεθος block = 5 • 8.192 = 40.960 bytes.

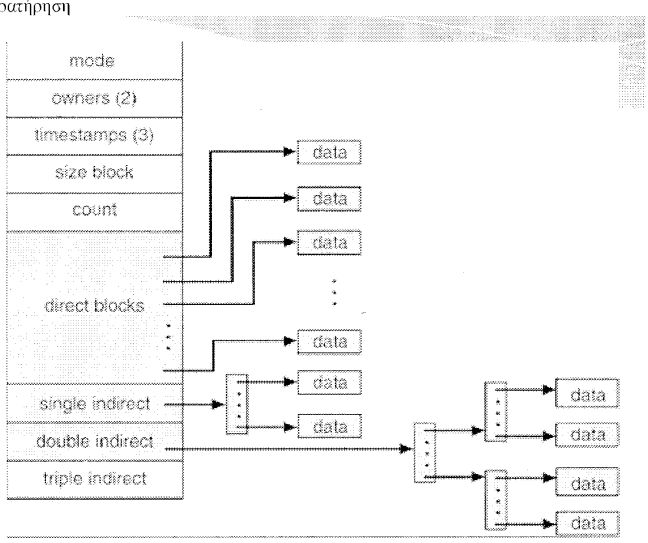
- Το πλήθος δεικτών είναι $\frac{8.192}{16}$ δείκτες και ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται μια απλή έμμεση καταχώριση είναι : $\frac{8.192}{16} \bullet 8.192 = 512 \bullet 8.192 = 4.194.304$ bytes. Επειδή έχουμε 2 απλές έμμεσες καταχωρήσεις ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από αυτές είναι: $2 \bullet 4.194.304 = 8.388.608$ bytes

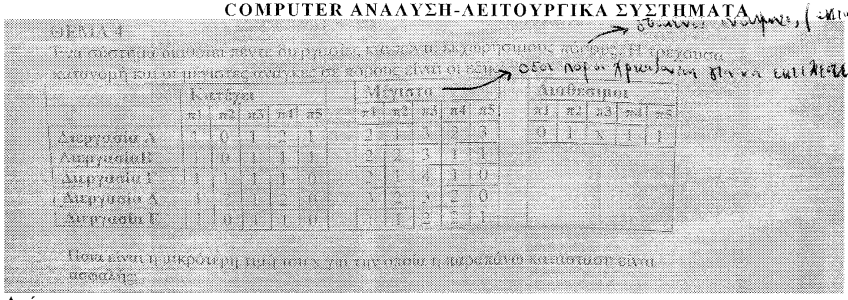
- Το πλήθος δεικτών είναι $\left(\frac{8.192}{16}\right)^2$ δείκτες και ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται μια διπλή έμμεση καταχώριση είναι: $\left(\frac{8.192}{16}\right)^2 \bullet 8.192 = 512^2 \bullet 8.192 = 2.147.483.648$ bytes. Επειδή έχουμε 4 διπλές έμμεσες καταχωρήσεις ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται από αυτές είναι: $4 \bullet 2.147.483.648 = 8.589.934.592$ bytes

- Το πλήθος δεικτών είναι $\left(\frac{8.192}{16}\right)^3$ δείκτες και ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται μια τριπλή έμμεση καταχώριση είναι: $\left(\frac{8.192}{16}\right)^3 \bullet 8.192 = 512^3 \bullet 8.192 = 1.099.511.627.776$ bytes. Επειδή έχουμε 3 τριπλές έμμεσες καταχωρήσεις ο μέγιστος αριθμός bytes που διευθυνσιοδοτείται είναι: $3 \bullet 1.099.511.627.776 = 3.298.534.883.328$ bytes

Το μέγιστο μέγεθος αρχείου είναι το άθροισμα όλων των προηγούμενων δηλ:
40.960 + 8.388.608 + 8.589.934.592 + 3.298.534.883.328 = 3.307.133.247.488 bytes περίπου 3 GB

Παρατήρηση





Απάντηση

ΘΕΜΑ 4

Υπόθεση 1

Έστω ότι το πλήθος αντιτύπων στον πόρο π3 είναι αρχικά Χ=0, οπότε η αρχική διαθεσιμότητα πόρων είναι η εξής:

Π1	Π2	Π3	Π4	Π5
0	1	0	1	1

Παρατηρώντας τον πίνακα με τις διεργασίες διαπιστώνουμε ότι με Χ=0 δεν μπορούμε να εκτελέσουμε καμία διεργασία με βάσει τις μέγιστες ανάγκες τους.

Υπόθεση 2

Έστω ότι το πλήθος αντιτύπων στον πόρο π3 είναι αρχικά Χ=1, οπότε η αρχική διαθεσιμότητα πόρων είναι η εξής:

Π1	Π2	Π3	Π4	Π5
0	1	1	1	1

Εκτελείται πρώτα η Διεργασία Ε. Όσο διαρκεί η εκτέλεση της Ε οι διαθέσιμοι πόροι είναι οι εξής:

Π1	Π2	Π3	Π4	Π5
0	0	0	0	0

Όταν τελειώσει η Διεργασία Ε αποδεσμεύει όλους τους πόρους που κατείχε (ΠΑΝΤΑ ΑΠΟΔΕΣΜΕΥΕΙ ΤΗ ΜΕΓΙΣΤΗ ΑΝΑΓΚΗ ΤΗΣ) και οι διαθέσιμοι πόροι είναι τώρα οι εξής:

Π1	Π2	Π3	Π4	Π5
1	1	2	2	1

Μετά δεν μπορεί να εκτελεστεί καμιά άλλη διεργασία με βάσει την υπάρχουσα διαθεσιμότητα πόρων

Βήμα 3

Έστω ότι το πλήθος αντιτύπων στον πόρο π3 είναι αρχικά Χ=2, οπότε η αρχική διαθεσιμότητα πόρων είναι η εξής:

Π1	Π2	Π3	Π4	Π5
0	1	2	1	1

Εκτελείται πρώτα η Διεργασία Ε. Όσο διαρκεί η εκτέλεση της Ε οι διαθέσιμοι πόροι είναι οι εξής:

Π1	Π2	Π3	Π4	Π5
0	0	1	0	0

Όταν τελειώσει η Διεργασία Ε αποδεσμεύει όλους τους πόρους που κατείχε (ΠΑΝΤΑ ΑΠΟΔΕΣΜΕΥΕΙ ΤΗ ΜΕΓΙΣΤΗ ΑΝΑΓΚΗ ΤΗΣ) και οι διαθέσιμοι πόροι είναι τώρα οι εξής:

Π1	Π2	Π3	Π4	Π5
1	1	3	2	1

Μετά εκτελείται πρώτα η Διεργασία Γ. Όσο διαρκεί η εκτέλεση της Γ οι διαθέσιμοι πόροι είναι οι εξής:

Π1	Π2	Π3	Π4	Π5
0	1	0	2	1

Όταν τελειώσει η Διεργασία Γ αποδεσμεύει όλους τους πόρους που κατείχε (ΠΑΝΤΑ ΑΠΟΔΕΣΜΕΥΕΙ ΤΗ ΜΕΓΙΣΤΗ ΑΝΑΓΚΗ ΤΗΣ) και οι διαθέσιμοι πόροι είναι τώρα οι εξής:

Π1	Π2	Π3	Π4	Π5
2	2	4	3	1

Μετά εκτελείται πρώτα η Διεργασία Δ. Όσο διαρκεί η εκτέλεση της Δ οι διαθέσιμοι πόροι είναι οι εξής:

Π1	Π2	Π3	Π4	Π5
0	2	2	3	1

Όταν τελειώσει η Διεργασία Δ αποδεσμεύει όλους τους πόρους που κατείχε (ΠΑΝΤΑ ΑΠΟΔΕΣΜΕΥΕΙ ΤΗ ΜΕΓΙΣΤΗ ΑΝΑΓΚΗ ΤΗΣ) και οι διαθέσιμοι πόροι είναι τώρα οι εξής:

Π1	Π2	Π3	Π4	Π5
3	4	5	5	1

Μετά εξυπηρετείται η Διεργασία Β. Όσο διαρκεί η εκτέλεση της Β οι διαθέσιμοι πόροι είναι οι εξής:

Π1	Π2	Π3	Π4	Π5
2	2	3	5	1

Όταν τελειώσει η Διεργασία Β αποδεσμεύει όλους τους πόρους που κατείχε (ΠΑΝΤΑ ΑΠΟΔΕΣΜΕΥΕΙ ΤΗ ΜΕΓΙΣΤΗ ΑΝΑΓΚΗ ΤΗΣ) και οι διαθέσιμοι πόροι είναι τώρα οι εξής:

Π1	Π2	Π3	Π4	Π5
4	4	6	6	2

Τέλος εκτελείται η Διεργασία Α οπότε το σύστημα με Χ=2 είναι σε ασφαλή κατάσταση γιατί υπάρχει να εξυπηρετηθούν όλες οι διεργασίες ακόμα και αν χρειαστούν τη μέγιστη ανάγκη τους σε πόρους..

ΘΕΜΑ 5

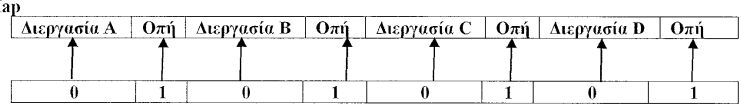
Απάντηση

ΘΕΜΑ 5

Α ερώτημα

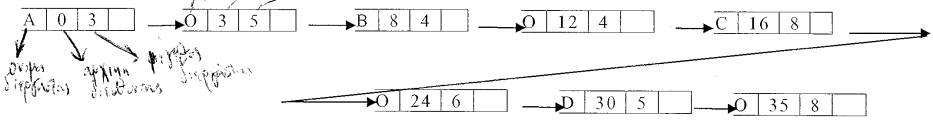
Παρατήρηση: Όταν θέλουμε να παρακολουθήσουμε τα κενά και τα κατειλημμένα διαμερίσματα μνήμης υπάρχουν 2 τρόποι: Ο Χάρτης Ψηφίων (Bitmap) όπου χρησιμοποιούμε ένα bit για το κάθε διαμέρισμα μνήμης με τιμή 1 αν είναι κενό (Οπή) ή 0 αν περιέχει διεργασία όπως στο ακόλουθο σχήμα και η Διασυνδεδεμένη Λίστα όπου χρησιμοποιούμε ένα κόμβο για κάθε διαμέρισμα μνήμης.

Bit Map



Η άσκηση εδώ ζητά την αναπαράσταση της μνήμης με το δεύτερο τρόπο με τη διασυνδεδεμένη λίστα. Η αναπαράσταση της μνήμης με τη λίστα φαίνεται στο ακόλουθο σχήμα:

Διασυνδεδεμένη Λίστα



Β ερώτημα

Αρχική Αναπαράσταση Μνήμης

Διεργασία/Οπή	Μέγεθος
A	3
O	5
B	4
O	4
C	8
O	6
D	5
O	8

First Fit

Αρχίζω από την αρχή της μνήμης και τοποθετώ τη διεργασία στην 1η οπή που τη χωράει. Η μνήμη μετά την τοποθέτηση της διεργασίας θα έχει την ακόλουθη μορφή:

Διεργασία/Οπή	Μέγεθος
A	3
E	4
O	1
B	4
O	4
C	8
O	6
D	5
O	8

Best Fit

Αρχίζω από την αρχή της μνήμης και τοποθετώ τη διεργασία στην οπή που τη χωράει καλύτερα (δηλ. αφήνει το λιγότερο κενό χώρο). Η μνήμη μετά την τοποθέτηση της διεργασίας θα έχει την ακόλουθη μορφή:

Διεργασία/Οπή	Μέγεθος
A	3
O	5
B	4
E	4
C	8
O	6
D	5
O	8

Worst Fit

Αρχίζω από την αρχή της μνήμης και τοποθετώ τη διεργασία στην μεγαλύτερη οπή που τη χωράει. Η μνήμη μετά την τοποθέτηση της διεργασίας θα έχει την ακόλουθη μορφή:

Διεργασία/Οπή	Μέγεθος
A	3
O	5
B	4
O	4
C	8
O	6
D	5
E	4
O	4

Next Fit

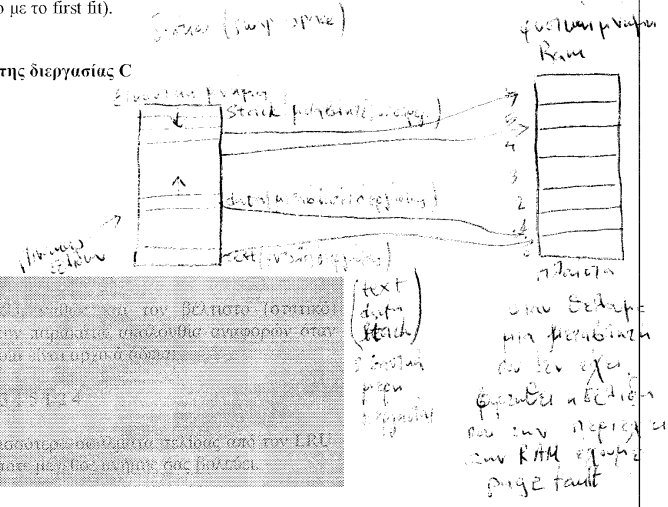
Μοιάζει με το first fit αλλά κάθε νέα αναζήτηση για κενό διαμέρισμα μνήμης γίνεται από εκεί που τελείωσε η προηγούμενη αναζήτηση και όχι από την αρχή της μνήμης (εδώ ίδιο με το first fit).

Γ ερώτημα

Αναπαράσταση Μνήμης μετά την αφαίρεση της διεργασίας C

Διεργασία/Οπή	Μέγεθος
A	3
O	5
B	4
O	18
D	5
O	8

Οι 3 οπές συνενώνονται στο τέλος σε μια οπή.



Απάντηση

ΘΕΜΑ 6

Α)**FIFO**

1	2	3	4	5	4	2	3	1	3	2	5	4	2	3	1	2	3	4	3	1	5	1	2	4
1	1	1	4	4	4	4	3	3	3	3	3	4	4	4	1	1	1	1	1	1	1	1	2	2
	2	2	2	5	5	5	5	1	1	1	1	1	2	2	2	2	2	4	4	4	4	4	4	4
		3	3	3	3	2	2	2	2	2	5	5	5	3	3	3	3	3	3	3	5	5	5	5
F	F	F	F	F	H	F	F	F	H	H	F	F	F	F	F	H	H	F	H	H	F	H	F	H

Σφάλματα σελίδας FIFO=16

LRU

1	2	3	4	5	4	2	3	1	3	2	5	4	2	3	1	2	3	4	3	1	5	1	2	4
1	1	1	4	4	4	4	4	1	1	1	5	5	5	3	3	3	3	3	3	3	3	3	2	2
	2	2	2	5	5	5	3	3	3	3	3	4	4	4	1	1	1	4	4	4	5	5	5	4
		3	3	3	3	2	2	2	2	2	2	2	2	2	2	2	2	2	2	1	1	1	1	1
F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**	**H**	**H**	**F**	**F**	**H**	**F**	**F**	**H**	**H**	**F**	**H**	**F**	**F**	**H**	**F**	**F**

Σφάλματα σελίδας LRU=17

ΒΕΛΤΙΣΤΟΣ

1	2	3	4	5	4	2	3	1	3	2	5	4	2	3	1	2	3	4	3	1	5	1	2	4
1	1	1	4	4	4	4	3	3	3	3	3	3	3	3	3	3	3	3	3	3	5	3	2	2
	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	4	4	4	4	5	5	4
		3	3	5	5	5	5	1	1	1	5	4	4	4	1	1	1	1	1	1	1	1	1	1
F	**F**	**F**	**F**	**F**	**H**	**H**	**F**	**F**	**H**	**H**	**F**	**F**	**H**	**H**	**F**	**H**	**H**	**F**	**H**	**H**	**F**	**H**	**F**	**F**

Σφάλματα Σελίδας ΒΕΛΤΙΣΤΟΥ=14

Β)

Απάντηση

a)Παράδειγμα όπου ο FIFO είναι καλύτερος από τον LRU (δηλ. έχει λιγότερα σφάλματα σελίδας)

FIFO=7 *Σφάλματα Σελίδας*

Σελίδες	1	2	3	4	5	4	2	1	5
	1	1	1	4	4	4	4	1	1
		2	2	2	5	5	5	5	5
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

LRU=8 *Σφάλματα Σελίδας*

Σελίδες	1	2	3	4	5	4	2	1	5
	1	1	1	4	4	4	4	4	5
		2	2	2	5	5	5	1	1
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

b)Παράδειγμα όπου ο LRU είναι καλύτερος από τον FIFO (δηλ. έχει λιγότερα σφάλματα σελίδας)

FIFO= 8 *Σφάλματα Σελίδας*

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	3	1
		2	2	2	5	5	5	5	4
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

LRU= 7 *Σφάλματα Σελίδας*

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	4	4
		2	2	2	5	5	5	3	3
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

ΘΕΜΑ 7

Ο αλγόριθμος τοποθέτησης θα επιλέξει να αντικαταστήσει δύο από τις οκτώ σελίδες στην εικόνα του δρόμου. Σε κάθε κύκλο το bit προφοράν άλλαξε.

0010100, 10100011, 00100001, 10101010, 11010111, 10010000

Ποια σελίδα θα αντικαταστήσει; Εξηγήστε με λεπτομέρεια την απάντησή σας.

Απάντηση

ΘΕΜΑ 7

Αρχικά

Σελίδα 1

0	0	0	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

0	0	0	0	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

0	0	0	0	0	0

Σελίδα 6

0	0	0	0	0	0

Σελίδα 7

0	0	0	0	0	0

Σελίδα 8

0	0	0	0	0	0

1ος κύκλος ρολογιού

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
0	0	1	0	0	1	0	0

Σελίδα 1

0	0	0	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

1	0	0	0	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

0	0	0	0	0	0

Σελίδα 6

1	0	0	0	0	0

Σελίδα 7

0	0	0	0	0	0

Σελίδα 8

0	0	0	0	0	0

2ος κύκλος ρολογιού-10100011

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
1	0	1	0	0	0	1	1

Σελίδα 1

1	0	0	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

1	1	0	0	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

0	0	0	0	0	0

Σελίδα 6

0	1	0	0	0	0

Σελίδα 7

1	0	0	0	0	0

Σελίδα 8

1	0	0	0	0	0

3ος κύκλος ρολογιού

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
0	0	1	0	0	1	0	1

Σελίδα 1

0	1	0	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

1	1	1	0	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

0	0	0	0	0	0

Σελίδα 6

1	0	1	0	0	0

Σελίδα 7

0	1	0	0	0	0

Σελίδα 8

1	1	0	0	0	0

4ος κύκλος ρολογιού

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
1	0	1	0	1	0	1	0

Σελίδα 1

1	0	1	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

1	1	1	1	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

1	0	0	0	0	0

Σελίδα 6

0	1	0	1	0	0

Σελίδα 7

1	0	1	0	0	0

Σελίδα 8

0	1	1	0	0	0

5ος κύκλος ρολογιού

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
1	1	1	0	1	0	1	1

Σελίδα 1

1	1	0	1	0	0

Σελίδα 2

1	0	0	0	0	0

Σελίδα 3

1	1	1	1	1	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

1	1	0	0	0	0

Σελίδα 6

0	0	1	0	1	0

Σελίδα 7

1	1	0	1	0	0

Σελίδα 8

1	0	1	1	0	0

6ος κύκλος ρολογιού

Σ1	Σ2	Σ3	Σ4	Σ5	Σ6	Σ7	Σ8
1	0	0	1	0	0	0	0

Σελίδα 1

1	1	1	0	1	0

Σελίδα 1 ακέραια τιμή μετρητή 58

Σελίδα 2

0	1	0	0	0	0

Σελίδα 2 ακέραια τιμή μετρητή 16

Σελίδα 3

0	1	1	1	1	1

Σελίδα 3 ακέραια τιμή μετρητή 31

Σελίδα 4

1	0	0	0	0	0

Σελίδα 4 ακέραια τιμή μετρητή 32

Σελίδα 5

0	1	1	0	0	0

Σελίδα 5 ακέραια τιμή μετρητή 24

Σελίδα 6

0	0	0	1	0	1

Σελίδα 6 ακέραια τιμή μετρητή 5

Σελίδα 7

0	1	1	0	1	0

Σελίδα 7 ακέραια τιμή μετρητή 26

Σελίδα 8

0	1	0	1	1	0

Σελίδα 8 ακέραια τιμή μετρητή 22

Σε κάθε κύκλο βάλτου - ρεύματος κάθε σελίδας αλλάζουν δεξιά κατά 1 θέση και στην κενή θέση αποθηκεύεται το Reference bit της σελίδας αυτής στο συγκεκριμένο κύκλο

Ο αλγόριθμος γήρανσης θα αντικαταστήσει τη σελίδα με το μικρότερο μετρητή. Αυτή είναι η σελίδα 6 με τιμή 5 στο μετρητή.

Γήρανση

ΘΕΜΑ 8

α. Περιγράψτε μια μαθηματική απολογισμού του βέλτιστου μεγέθους σελίδας ώστε να ελαχιστοποιείται η συνολική επιβάρυνση χώρου για την αποθήκευση του πίνακα σελίδων και της εσωτερικής κατάτμησης.

β. Ποιο είναι το βέλτιστο μέγεθος σελίδας εάν το μέγεθος διεργασίας είναι 1MB και ο χώρος ανά σελίδα στον πίνακα σελίδων είναι 128 bytes;

Απάντηση

ΘΕΜΑ 8

α)Έστω ότι το μέσο μέγεθος διεργασίας είναι **μ byte** και το μέγεθος σελίδας είναι **σ byte**. Επιπλέον ας υποθέσουμε ότι κάθε καταχώριση σελίδας απαιτεί **κ bytes**. Ο αριθμός των σελίδων που χρειάζονται για κάθε διεργασία είναι κατά προσέγγιση **μ/σ** και ο χώρος που θα καταληφθεί στον πίνακα σελίδων θα είναι **μκ/σ bytes**. Η μνήμη που χάνεται στην τελευταία σελίδα λόγω της εσωτερικής κατάτμησης είναι **σ/2**. Επομένως η συνολική επιβάρυνση λόγω του πίνακα σελίδων και της εσωτερικής κατάτμησης δίνεται από το άθροισμα δύο όρων: **Επιβάρυνση = μκ/σ +σ/2**

Ο 1ος όρος **μκ/σ** (μέγεθος του πίνακα σελίδων) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μικρό. Ο 2ος όρος **σ/2** (η εσωτερική κατάτμηση) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μεγάλο. Το βέλτιστο μέγεθος πρέπει να βρίσκεται κάπου ανάμεσα. Παίρνουμε την 1η παράγωγο **ως προς σ** και εξισώνουμε με το μηδέν οπότε προκύπτει η εξίσωση:**-μκ/σ^2 +1/2=0**

Από την εξίσωση αυτή μπορούμε να κατασκευάσουμε ένα τύπο που δίνει το βέλτιστο μέγεθος σελίδας (λαμβάνοντας υπόψη μόνο τη μνήμη που χάνεται από την κατάτμηση και το μέγεθος του πίνακα σελίδων). Το αποτέλεσμα είναι: $\sigma = \sqrt{2\mu\kappa}$

β)Για μ=1 Mbyte και k=128 byte για κάθε καταχώριση στον πίνακα σελίδων, το βέλτιστο μέγεθος σελίδας είναι $\sigma = \sqrt{2 \bullet 10^6 \bullet 128}$ Bytes

ΘΕΩΡΙΑ Β ΟΜΑΔΑΣ Φεβρουάριος 2014

ΘΕΜΑ 1

(α) Παρουσιάστε τις έννοιες της συμβολικής σύνδεσης (symbolic linking) και της παραδοσιακής (πραγματικής-hard) σύνδεσης σε ένα σύστημα διαχείρισης αρχείων και καταγράψτε τα θετικά και τα αρνητικά τους.

(β) Εξηγήστε ποιες οι διαφορές διεργασιών και νημάτων.

Λύση

(α)

Ο σύνδεσμος (link) είναι μια τεχνική που επιτρέπει σε ένα αρχείο να εμφανίζεται σε περισσότερους από ένα καταλόγους. Η κλήση συστήματος (link) στο Unix καθορίζει ένα υπάρχον αρχείο και ένα όνομα διαδρομής και δημιουργεί ένα σύνδεσμο από το υπάρχον αρχείο προς το όνομα που καθορίζει η διαδρομή. Με τον τρόπο αυτό το ίδιο αρχείο μπορεί να εμφανίζεται σε περισσότερους καταλόγους. Οι σύνδεσμοι αυτού του είδους αυξάνουν το μετρητή του i-node του αρχείου για να παρακολουθούν τον αριθμό των καταλόγων που περιέχουν το αρχείο και ονομάζονται **πραγματικοί σύνδεσμοι (hard links).**

Μια παραλλαγή της ιδέας των συνδέσμων είναι ο **συμβολικός σύνδεσμος** (symbolic link). Αντί να έχουμε δύο ονόματα που δείχνουν στην ίδια εσωτερική δομή δεδομένων που αναπαριστά ένα αρχείο, να μπορούμε να δημιουργήσουμε ένα όνομα το οποίο δείχνει σε ένα μικροσκοπικό αρχείο που κατονομάζει ένα άλλο αρχείο. Όταν χρησιμοποιείται το πρώτο αρχείο το σύστημα αρχείων ακολουθεί τη διαδρομή και βρίσκει το όνομα στο τέλος της. Στη συνέχεια από την αρχή τη διαδικασία αναζήτησης χρησιμοποιώντας το νέο όνομα. Οι συμβολικοί σύνδεσμοι έχουν το πλεονέκτημα ότι μπορούν να ξεπερνούν τα όρια ενός δίσκου και ακόμη και να δίνουν ονόματα σε αρχεία απομακρυσμένων υπολογιστών. Η υλοποίηση τους είναι ελαφρώς λιγότερο αποδοτική από τους πραγματικούς συνδέσμους.

(β)

ΔΙΑΦΟΡΕΣ ΔΙΕΡΓΑΣΙΩΝ-ΝΗΜΑΤΩΝ

Διεργασία είναι μια εφαρμογή που εκτελείται και διαχειρίζεται από το λειτουργικό σύστημα

- ✓ Είναι Ανεξάρτητη μονάδα εκτέλεσης και περιλαμβάνει:
  - Πληροφορίες κατάστασης (State information)
  - Ιδεατό χώρο μνήμης (Virtual memory)
- ✓ Η Επικοινωνία μεταξύ διεργασιών με σαφώς προκαθορισμένους τρόπους

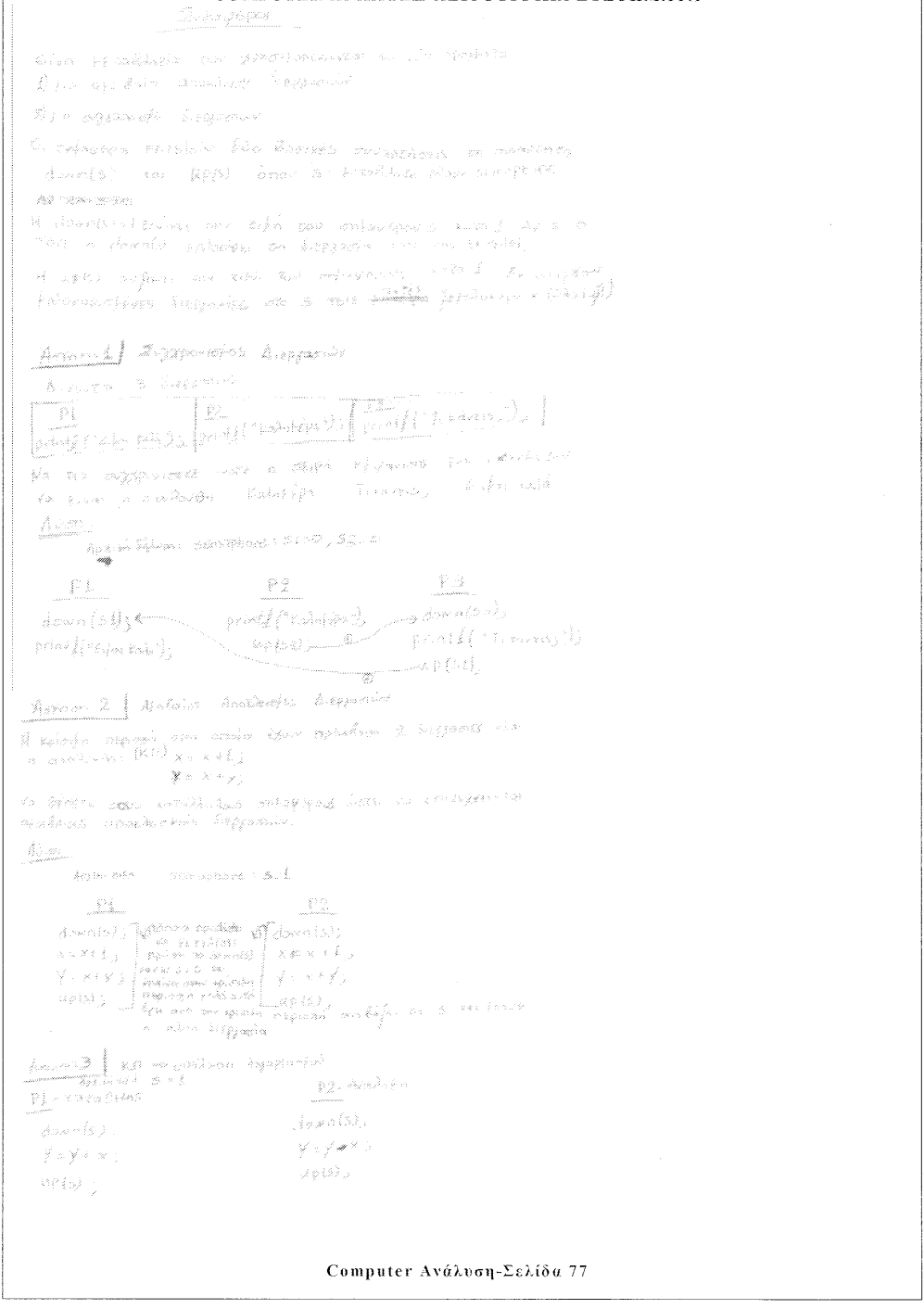
Νήμα είναι ένα μονοπάτι εκτέλεσης εντολών εντός μιας διεργασίας

- ✓ Κάθε διεργασία έχει τουλάχιστον ένα νήμα
- ✓ Όλα τα νήματα μοιράζονται τους πόρους της διεργασίας
  - Πολύ σημαντικός πόρος είναι ο χώρος διευθύνσεων ιδεατής μνήμης
  - Όλα τα νήματα έχουν άμεση επικοινωνία μεταξύ τους και έχουν άμεση πρόσβαση σε όλες τις μεταβλητές
  - Το νήμα μπορεί να υποστεί διαχείριση:
    - ✓ είτε από το λειτουργικό σύστημα (Kernel-level thread)
    - ✓ είτε στο χώρο διευθύνσεων μνήμης του χρήστη (User-level thread). Αυτό γίνεται συνήθως σε μια βιβλιοθήκη νημάτων

Επιπλέον Διαφορές Διεργασίας και Νήματος είναι οι ακόλουθες:

Τα νήματα έχουν πολλές ομοιότητες με τις διεργασίες αλλά και δύο διαφορές:

- Έχουν όλα πρόσβαση σε μια κοινή περιοχή της μνήμης
- Η διαδικασία εναλλαγής τους στην ΚΜΕ είναι απλούστερη από την αντίστοιχη διεδικασία εναλλαγής διαδικασιών.



6 Θέματα Ιούνιος 2014

ΘΕΜΑ 1

(i) Εξηγείστε την έννοια των σημαφόρων και αναφέρετε παραδείγματα εφαρμογής τους

(ii) Εξηγείστε την έννοια των ιεραρχικών καταλόγων και καταγράψτε τις βασικές λειτουργίες/υπηρεσίες που προσφέρει σε ένα σύστημα καταλόγων

Απάντηση

Οι σημαφόροι είναι ένας τύπος μεταβλητών που προσφέρουν **συγχρονισμό** και **αμοιβαίο αποκλεισμό** διεργασιών. Στην ουσία ένας σημαφόρος "μετράει" όλες τις κλήσεις αφύπνισης για ένα process. Στους σημαφόρους εφαρμόζονται 2 βασικές πράξεις: Οι UP και DOWN

DOWN(s) : 1 ατομική ενέργεια **(atomic)**

```
{
   if s = 0 then sleep();
   else s = s – 1;
}
```

UP(s) : 1 ατομική ενέργεια **(atomic)**

```
{
    s: = s + 1;
}
```

Αν μετά την εκτέλεση του UP(s) υπάρχει κάποιο process που "κοιμάται" τότε το σύστημα διαλέγει κάποιο από τα μπλοκαρισμένα process και το ξυπνάει.

(ii)

Για την οργάνωση των αρχείων πολλά λειτουργικά συστήματα υλοποιούν ιεραρχική δομή καταλόγων (*tree directory structure*). Η πρόσβαση σε ένα αρχείο μπορεί να γίνει είτε με:

- απόλυτο όνομα διαδρομής (*absolute file path*) ή με
- σχετικό όνομα διαδρομής (*relative file path*) ως προς τον τρέχοντα κατάλογο (*current directory*)

Με τους ιεραρχικούς καταλόγους μπορούμε να οργανώσουμε τις πληροφορίες μας (αρχεία και καταλόγους) σε μια δεντρική δομή και να τις προσπελάσουμε εύκολα και γρήγορα.

Η πρόσβαση στους καταλόγους γίνεται μέσω κλήσεων στο λειτουργικό σύστημα. Οι σημαντικότερες από αυτές είναι:

- delete → Διαγραφή ενός αρχείου από έναν κατάλογο
- mkdir → Δημιουργία ενός καταλόγου
- rmdir → Διαγραφή ενός καταλόγου
- opendir → Αρχή πρόσβασης στα στοιχεία ενός καταλόγου
- readdir →Ανάγνωση στοιχείων από έναν κατάλογο
- closedir →Τέλος της πρόσβασης στον κατάλογο
- link →Σύνδεση δύο αρχείων μεταξύ τους

ΘΕΜΑ 2

(i)Ένα σύστημα διαθέτει 2 διεργασίες και 3 πανομοιότυπους πόρους. Το μέγιστο που χρειάζεται κάθε διεργασία είναι 2 πόροι. Υπάρχει πιθανότητα να συμβεί αδιέξοδο; Εξηγείστε την απάντηση σας

(ii) Ένας υπολογιστής έχει έξι μονάδες ταινίας και δ διεργασίες που τις διεκδικούν. Κάθε διεργασία μπορεί να χρειαστεί το πολύ 2 μονάδες ταινίας. Για ποιες τιμές του δ αποκλείεται η εμφάνιση αδιεξόδου στο σύστημα;

Απάντηση

(i) **Ισχύει το εξής Θεώρημα**: Έστω ένα σύστημα με P διεργασίες και R πανομοιότυπους πόρους. Αν κάθε διεργασία απαιτεί κατά μέγιστο 2 μονάδες πόρων, ότι δεν μπορεί να συμβεί αδιέξοδο μόνον όταν ισχύει η συνθήκη P<=R-1.

Απόδειξη Θεωρήματος

Αν σε κάποια χρονική στιγμή οι P διεργασίες έχουν δεσμεύσει η κάθε μια από μια μονάδα του πόρου R και υπάρχει μια ακόμα ελεύθερη-διαθέσιμη μονάδα του πόρου, τότε μία από τις P διεργασίες θα τη δεσμεύσει, θα τη χρησιμοποιήσει, θα ολοκληρώσει την εκτέλεση της και θα επιστρέψει τις 2 μονάδες πόρου στο σύστημα ώστε να συνεχιστεί και η εκτέλεση των υπολοίπων διεργασιών. Δηλαδή σε αυτήν την οριακή περίπτωση θα πρέπει να ισχύει: P+1=R ⇒ P=R-1

Για προφανείς λόγους οποιαδήποτε τιμή πλήθους διεργασιών P < R-1 δεν μπορεί να δημιουργεί αδιέξοδο μια και θα υπάρχουν διαθέσιμες περισσότερες μονάδες του πόρου. Άρα τελικά οι δύο σχέσεις ενσωματώνονται σε μία ως εξής: P<=R-1

Άρα για P=2 και R=3 και επειδή το μέγιστο που χρειάζεται μια διεργασία είναι 2 πόροι, η συνθήκη P=2<=3-1 **ΙΣΧΥΕΙ ΑΡΑ ΔΕΝ ΘΑ ΕΜΦΑΝΙΣΤΕΙ ΑΔΙΕΞΟΔΟ**

(ii)

Ισχύει το εξής Θεώρημα: Σε ένα σύστημα υπάρχουν N ενεργές διεργασίες που διαμοιράζονται M μονάδες ενός επαναχρησιμοποιήσιμου πόρου. Κάθε διεργασία μπορεί να απαιτήσει το πολύ 3 μονάδες του πόρου R. Να βρείτε ποια σχέση πρέπει να έχουν οι παράμετροι N και M ώστε να μην υπάρχει κίνδυνος αδιεξόδου.

Απόδειξη Θεωρήματος

Αν σε κάποια χρονική στιγμή οι N ενεργές διεργασίες έχουν ήδη δεσμεύσει η κάθε μία από 2 μονάδες του πόρου R τότε θα είναι δεσμευμένες 2N μονάδες του πόρου και πρέπει να υπάρχει τουλάχιστον ακόμη μία μονάδα διαθέσιμη ώστε μία από τις N διεργασίες να την αποκτήσει, να τη χρησιμοποιήσει, να ολοκληρώσει την εκτέλεσή της και στη συνέχεια να επιστρέψει τις 3 μονάδες στο σύστημα ώστε να χρησιμοποιηθούν από άλλες διεργασίες που περιμένουν να απελευθερωθούν πόροι για να μπορέσουν να συνεχίσουν την εκτέλεσή τους. Στην οριακή αυτή περίπτωση λοιπόν θα πρέπει να ισχύει :

$$2N+1=M$$

Οποιαδήποτε τιμή M μεγαλύτερη από το όριο που υπολογίστηκε παραπάνω, θα εξασφαλίζει, κατά μείζονα λόγο την αποφυγή του αδιεξόδου δηλαδή τελικά :

$$M >= 2N+1$$

Επαναδιατυπώνουμε το προηγούμενο θεώρημα ως εξής: Έστω ότι στο σύστημα υπάρχουν N=δ ενεργές διεργασίες που διαμοιράζονται M=6 μονάδες ενός επαναχρησιμοποιήσιμου πόρου. Κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 2 μονάδες του πόρου. Αν σε κάποια χρονική στιγμή οι N=δ ενεργές διεργασίες έχουν ήδη δεσμεύσει η κάθε μία από 1 μονάδα του πόρου τότε θα είναι δεσμευμένες N=δ μονάδες του πόρου και πρέπει να υπάρχει τουλάχιστον ακόμη μία μονάδα διαθέσιμη ώστε μία από τις N=δ διεργασίες να την αποκτήσει, να τη χρησιμοποιήσει, να ολοκληρώσει την εκτέλεσή της και στη συνέχεια να επιστρέψει τις 2 μονάδες στο σύστημα ώστε να χρησιμοποιηθούν από άλλες διεργασίες που περιμένουν να απελευθερωθούν πόροι για να μπορέσουν να συνεχίσουν την εκτέλεσή τους. Στην οριακή αυτή περίπτωση λοιπόν θα πρέπει να ισχύει:**δ+1=6**

Οποιαδήποτε τιμή μεγαλύτερη από το όριο αυτό (το 6), θα εξασφαλίζει την αποφυγή του αδιεξόδου δηλαδή τελικά : **6 >=δ+1**

ΘΕΜΑ 3

(i)Πόσες λειτουργίες δίσκου χρειάζονται για να μεταφερθεί ο κόμβος i του αρχείου **/usr/ast/courses/os/handout.t**. Υποθέστε ότι ο κόμβος i που αντιστοιχεί στο βασικό κατάλογο βρίσκεται στη μνήμη αλλά δε βρίσκεται εκεί τίποτε άλλο σχετικό με τη διαδρομή. Υποθέστε επίσης ότι όλοι οι κατάλογοι χωρούν σε ένα μπλοκ δίσκου.

(ii) Η απόδοση ενός συστήματος αρχείων εξαρτάται σε ένα μεγάλο βαθμό από το ποσοστό ευστοχίας της κρυφής μνήμης (το ποσοστό των μπλοκ που βρέθηκαν στην κρυφή μνήμη). Αν χρειάζεται 1 msec για να ικανοποιηθεί μια αίτηση από την κρυφή μνήμη, αλλά 40 msec για να ικανοποιηθεί μια αίτηση όταν χρειάζεται ανάγνωση από το δίσκο, δώστε ένα μαθηματικό τύπο για το μέσο χρόνο που απαιτείται για να ικανοποιηθεί μια αίτηση αν το ποσοστό ευστοχίας είναι π.

Απάντηση

(i)

Σε ένα σύστημα UNIX, πόσες αναγνώσεις από το δίσκο πρέπει να γίνουν για να διαβαστεί ο κόμβος-i για το αρχείο `/usr/ast/courses/os/handout.t`;. Υποθέστε ότι ο κόμβος-i για τον κατάλογο της ρίζας βρίσκεται στη μνήμη. Κανένας από τους άλλους κόμβους-i καταλόγων της διαδρομής δεν βρίσκεται στην μνήμη.

Απάντηση

Έχουμε τις παρακάτω αναγνώσεις:

1. κατάλογος για / (ο κόμβος-i δεν χρειάζεται να διαβαστεί για τη ρίζα, αφού σύμφωνα με την εκφώνηση είναι στη μνήμη ήδη).
2. κόμβος-i για /usr
3. κατάλογος για /usr
4. κόμβος-i για /usr/ast
5. κατάλογος για /usr/ast
6. κόμβος-i για /usr/ast/courses
7. κατάλογος για /usr/ast/courses
8. κόμβος-i για /usr/ast/courses/os
9. κατάλογος για /usr/ast/courses/os
10. κόμβος-i για /usr/ast/courses/os/handout.t

Συνολικά χρειάζονται 10 αναγνώσεις από το δίσκο.

(ii)

Ο μέσος χρόνος που απαιτείται για να ικανοποιηθεί μια αίτηση είναι: $t_\mu = \pi \bullet t_c + (1-\pi) \bullet (t_c + t_p)$ όπου:

$t_\mu \rightarrow$ο μέσος χρόνος προσπέλασης
$t_c \rightarrow$ο χρόνος προσπέλασης στην κρυφή μνήμη
$t_p \rightarrow$ο χρόνος προσπέλασης στο σκληρό δίσκο
$\pi \rightarrow$η πιθανότητα το μπλοκ να βρίσκεται στην κρυφή μνήμη
$1-\pi \rightarrow$η πιθανότητα το μπλοκ να μην βρίσκεται στην κρυφή μνήμη

Εδώ t_p=40 msec, t_c=1 msec

Παρατήρηση

Στη μνήμη cache αποθηκεύουμε blocks του δίσκου.

ΘΕΜΑ 4

(ii)

Ισχύει το εξής Θεώρημα: Σε ένα σύστημα υπάρχουν N σε πλήθος ενεργές διεργασίες που διαμοιράζονται M μονάδες ενός επαναχρησιμοποιήσιμου πόρου. Κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 3 μονάδες του πόρου R. Να βρείτε ποια σχέση πρέπει να έχουν οι παράμετροι N και M ώστε να μην υπάρχει κίνδυνος αδιεξόδου.

Απόδειξη Θεωρήματος

Αν σε κάποια χρονική στιγμή οι N ενεργές διεργασίες έχουν ήδη δεσμεύσει η κάθε μία από 2 μονάδες του πόρου R τότε θα είναι δεσμευμένες 2N μονάδες του πόρου και πρέπει να υπάρχει τουλάχιστον ακόμη μία μονάδα διαθέσιμη ώστε μία από τις N διεργασίες να την αποκτήσει, να τη χρησιμοποιήσει, να ολοκληρώσει την εκτέλεσή της και στη συνέχεια να επιστρέψει τις 3 μονάδες στο σύστημα ώστε να χρησιμοποιηθούν από άλλες διεργασίες που περιμένουν να απελευθερωθούν πόροι για να μπορέσουν να συνεχίσουν την εκτέλεσή τους. Στην οριακή αυτή περίπτωση λοιπόν θα πρέπει να ισχύει :

$$2N+1=M$$

Οποιαδήποτε τιμή M μεγαλύτερη από το όριο που υπολογίστηκε παραπάνω, θα εξασφαλίζει, κατά μείζονα λόγο την αποφυγή του αδιεξόδου δηλαδή τελικά :

$$M >= 2N+1$$

Επαναδιατυπώνουμε το προηγούμενο θεώρημα ως εξής: Έστω ότι στο σύστημα υπάρχουν N=δ σε πλήθος ενεργές διεργασίες που διαμοιράζονται M=6 μονάδες ενός επαναχρησιμοποιήσιμου πόρου. Κάθε διεργασία μπορεί να απαιτήσει κατά μέγιστο 2 μονάδες του πόρου. Αν σε κάποια χρονική στιγμή οι N=δ ενεργές διεργασίες έχουν ήδη δεσμεύσει η κάθε μία από 1 μονάδα του πόρου τότε θα είναι δεσμευμένες N=δ μονάδες του πόρου και πρέπει να υπάρχει τουλάχιστον ακόμη μία μονάδα διαθέσιμη ώστε μία από τις N=δ διεργασίες να την αποκτήσει, να τη χρησιμοποιήσει, να ολοκληρώσει την εκτέλεσή της και στη συνέχεια να επιστρέψει τις 2 μονάδες στο σύστημα ώστε να χρησιμοποιηθούν από άλλες διεργασίες που περιμένουν να απελευθερωθούν πόροι για να μπορέσουν να συνεχίσουν την εκτέλεσή τους. Στην οριακή αυτή περίπτωση λοιπόν θα πρέπει να ισχύει :

$$\delta+1=6$$

Οποιαδήποτε τιμή μεγαλύτερη από το όριο που υπολογίστηκε παραπάνω, θα εξασφαλίζει, κατά μείζονα λόγο την αποφυγή του αδιεξόδου δηλαδή τελικά : $6 >= \delta+1$

ΘΕΜΑ 3

(i)Πόσες λειτουργίες δίσκου χρειάζονται για να μεταφερθεί ο κόμβος i του αρχείου **/usr/ast/courses/os/handout.t**. Υποθέστε ότι ο κόμβος i που αντιστοιχεί στο βασικό κατάλογο βρίσκεται στη μνήμη αλλά δεν βρίσκεται εκεί τίποτε άλλο σχετικό με τη διαδρομή. Υποθέστε επίσης ότι όλοι οι κατάλογοι χωρούν σε ένα μπλοκ δίσκου.

(ii) Η απόδοση ενός συστήματος αρχείων εξαρτάται σε ένα μεγάλο βαθμό από το ποσοστό ευστοχίας της κρυφής μνήμης (το ποσοστό των μπλοκ που βρέθηκαν στην κρυφή μνήμη). Αν χρειάζεται 1 msec για να ικανοποιηθεί μια αίτηση από την κρυφή μνήμη, αλλά 40 msec για να ικανοποιηθεί μια αίτηση όταν χρειάζεται ανάγνωση από το δίσκο, δώστε ένα μαθηματικό τύπο για το μέσο χρόνο που απαιτείται για να ικανοποιηθεί μι αίτηση αν το ποσοστό ευστοχίας είναι π.

Απάντηση

(i)

Μετάφραση Ονόματος (Pathname translation) είναι Η διαδικασία που δέχεται σαν είσοδο ένα pathname και παράγει σαν έξοδο το inode# του αρχείου. Εδώ το μονοπάτι είναι το **/usr/ast/courses/os/handout.t**

Η διαδικασία αυτή έχει ως εξής:

1.
   a) Το inode για το root directory "/" ανακτάται από τον δίσκο. (Αυτό το inode βρίσκεται σε μια γνωστή και μη ενημερώσιμη διεύθυνση στο δίσκο).
   b) Από το inode του root βρίσκονται τα disk blocks (που περιέχουν τα dir. entries), ανακτώνται και ψάχνονται για την εγγραφή "usr". Όταν βρίσκεται η εγγραφή για το "usr" τότε βρίσκεται και το inode# του usr.

2.
   a) Το inode για το "usr" ανακτάται απ' το δίσκο.
   b) Τα disk blocks του dir file "usr" ανακτώνται και ψάχνονται για την εγγραφή "ast"

3.
   a) Το inode ανακτάται από το δίσκο
   b) Τα disk blocks του dir file "ast" ανακτώνται και ψάχνονται για την εγγραφή "courses"

4.
   a) Το inode ανακτάται από το δίσκο
   b) Τα disk blocks του dir file "courses" ανακτώνται και ψάχνονται για την εγγραφή "os"

5.
   a) Τέλος το inode & disk blocks του "os" ανακτώνται και ψάχνονται για την εγγραφή "handout.t"
   b) Έτσι βρίσκεται το inode# για το αρχείο «handout.t» το οποίο ανακτάται και τοποθετείται στη μνήμη (σ' ένα cache ειδικό για inodes που λέγεται inode table)
   c) Αφού τοποθετηθεί το inode στο inode table του πυρήνα ο file descriptor πίνακας του process ενημερώνεται.

d) Μια εγγραφή του χρησιμοποιείται για ένα δείκτη ακολουθώντας τον οποίο μπορεί ο kernel να βρει το inode στο inode table

e) Τέλος επιστρέφεται ο δείκτης της εγγραφής του file descriptor πίνακα...

(ii)

Ο τύπος που δίνει το μέσο χρόνο που απαιτείται για να ικανοποιηθεί μια αίτηση στην κρυφή μνήμη δίνεται από τον τύπο:

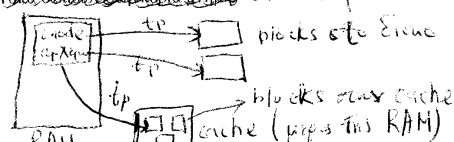
$$t_\mu = \pi \bullet t_c + (1-\pi) \bullet (t_c + t_p) \text{ όπου}$$

$t_\mu \rightarrow$ο μέσος χρόνος προσπέλασης στην κρυφή μνήμη

$t_c \rightarrow$ο χρόνος προσπέλασης στην κρυφή (συχετιστική) μνήμη

$t_p \rightarrow$ο χρόνος προσπέλασης στην κύρια μνήμη (όταν μια σελίδα χρειάζεται από την κύρια μνήμη)

Εδώ t_p=40 msec, t_c=1 msec



Παρατήρηση

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

- 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

ΘΕΜΑ 4

Εκτελέστε τους αλγορίθμους FIFO, LRU καθώς και το βέλτιστο (στατικό) αλγόριθμο αντικατάσταση σελίδων για την παρκάτω ακολουθίας αναφορών όταν είναι διαθέσιμα 3 πλαίσια μνήμης τα οποία αρχικά είναι άδεια:

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

Πόσα σφάλματα σελίδας κάνει ο καθένας. Δώστε παράδειγμα που ο FIFO κάνει λιγότερα σφάλματα σελίδας από τον LRU και αντίστροφα. Χρησιμοποιήστε οποιοδήποτε μέγεθος μνήμης σας βολεύει

Απάντηση

(i)

FIFO

2	**3**	**4**	**1**	**4**	**2**	**3**	**4**	**3**	**1**	**2**	**4**
2	2	2	1	1	1	1	4	4	4	4	4
	3	3	3	3	2	2	2	2	1	1	1
		4	4	4	4	3	3	3	3	2	2
F	**F**	**F**	**F**	**H**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

LRU

2	**3**	**4**	**1**	**4**	**2**	**3**	**4**	**3**	**1**	**2**	**4**
2	2	2	1	1	1	3	3	3	3	3	4
	3	3	3	3	2	2	2	2	1	1	1
		4	4	4	4	4	4	4	4	2	2
F	**F**	**F**	**F**	**H**	**F**	**F**	**H**	**H**	**F**	**F**	**F**

OPT

2	**3**	**4**	**1**	**4**	**2**	**3**	**4**	**3**	**1**	**2**	**4**
2	2	2	2	2	2	2	2	2	2	2	4
	3	3	1	1	1	3	3	3	3	3	3
		4	4	4	4	4	4	4	1	1	1
F	**F**	**F**	**F**	**H**	**H**	**F**	**H**	**H**	**F**	**H**	**F**

(ii)

a)Παράδειγμα όπου ο FIFO είναι καλύτερος από τον LRU (δηλ.. έχει λιγότερα σφάλματα σελίδας)

FIFO=7 P.F.

Σελίδες	1	2	3	4	5	4	2	1	5
	1	1	1	4	4	4	4	1	1
		2	2	2	5	5	5	5	5
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

LRU=8 P.F.

Σελίδες	1	2	3	4	5	4	2	1	5
	1	1	1	4	4	4	4	4	5
		2	2	2	5	5	5	1	1
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

b)Παράδειγμα όπου ο LRU είναι καλύτερος από τον FIFO (δηλ.. έχει λιγότερα σφάλματα σελίδας)

FIFO= 8 P.F.

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	3	1
		2	2	2	5	5	5	5	4
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**F**

LRU= 7 P.F.

Σελίδες	1	2	3	4	5	4	2	3	4
	1	1	1	4	4	4	4	4	4
		2	2	2	5	5	5	3	3
			3	3	3	3	2	2	2
	F	**F**	**F**	**F**	**F**	**H**	**F**	**F**	**H**

ΘΕΜΑ 5

Ο αλγόριθμος γήρανσης θα επιλέξει να αντικαταστήσει δύο από τις 8 σελίδες στον έκτο κύκλο ρολογιού. Σε κάθε κύκλο ρολογιού τα bits αναφορών είναι:

01111011
01011011
01110111
01011000
11101100
01010011

Λύση

Αρχικά

Σελίδα 1

0	0	0	0	0	0

Σελίδα 2

0	0	0	0	0	0

Σελίδα 3

0	0	0	0	0	0

Σελίδα 4

0	0	0	0	0	0

Σελίδα 5

0	0	0	0	0	0

Σελίδα 6

0	0	0	0	0	0

Σελίδα 7

0	0	0	0	0	0

Σελίδα 8

0	0	0	0	0	0

Στον 6ο κύκλο ρολογιού οι μετρητές είναι:

Σελίδα 1

0	1	0	0	0	0	Ακέραια τιμή 16

Σελίδα 2

1	1	1	1	1	1	Ακέραια τιμή 63

Σελίδα 3

0	1	0	1	0	1	Ακέραια τιμή 21

Σελίδα 4

1	0	1	1	1	1	Ακέραια τιμή 47

Σελίδα 5

0	1	1	0	1	1	Ακέραια τιμή 27

Σελίδα 6

0	1	0	1	0	0	Ακέραια τιμή 20

Σελίδα 7

1	0	0	1	1	1	Ακέραια τιμή 39

Σελίδα 8

1	0	0	1	1	1	Ακέραια τιμή 39

Ο *αλγόριθμος γήρανσης* θα αντικαταστήσει τις 2 σελίδες με το μικρότερο μετρητή. Αυτές είναι οι σελίδα 1 και 6.

ΘΕΜΑ 6

α)Έστω ότι το μέσο μέγεθος διεργασίας είναι **μ byte** και το μέγεθος σελίδας είναι **σ byte**. Επιπλέον ας υποθέσουμε ότι κάθε καταχώριση σελίδας απαιτεί **κ bytes**. Ο αριθμός των σελίδων που χρειάζονται για κάθε διεργασία είναι κατά προσέγγιση **μ/σ** και ο χώρος που θα καταληφθεί στον πίνακα σελίδων θα είναι **μκ/σ bytes**. Η μνήμη που χάνεται στην τελευταία σελίδα λόγω της εσωτερικής κατάτμησης είναι **σ/2**. Επομένως η συνολική επιβάρυνση λόγω του πίνακα σελίδων και της εσωτερικής κατάτμησης δίνεται από το άθροισμα δύο όρων:

Επιβάρυνση = μκ/σ +σ/2

Ο 1ος όρος **μκ/σ** (το μέγεθος του πίνακα σελίδων) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μικρό. Ο 2ος όρος **σ/2** (η εσωτερική κατάτμηση) είναι μεγάλος όταν το μέγεθος της σελίδας είναι μεγάλο. Το βέλτιστο μέγεθος πρέπει να βρίσκεται κάπου ανάμεσα. Παίρνουμε την 1η παράγωγο **ως προς σ** και εξισώνουμε με το μηδέν οπότε προκύπτει η εξίσωση:

$-\mu\kappa/\sigma^2 +1/2=0$

Από την εξίσωση αυτή μπορούμε να κατασκευάσουμε ένα τύπο που δίνει το βέλτιστο μέγεθος σελίδας (λαμβάνοντας υπόψη μόνο τη μνήμη που χάνεται από την κατάτμηση και το μέγεθος του πίνακα σελίδων). Το αποτέλεσμα είναι: $\sigma = \sqrt{2\mu\kappa}$

β)Για μ=1 Mbyte και k=128 byte για κάθε καταχώριση στον πίνακα σελίδων, το βέλτιστο μέγεθος σελίδας είναι $\sigma = \sqrt{2 \bullet 10^6 \bullet 128}$ Bytes