SCHEDULING

ΑΣΚΗΣΗ 1

Θεωρήστε τις ακόλουθες διεργασίες, οι οποίες εκτελούν αποκλειστικά υπολογισμό χωρίς /διάμεσο I/O, σφάλματα σελίδων, κλήσεις συστήματος κλπ.:

Διεργασία	Χρόνος Άφιξης	Χρόνος Εκτέλεσης	Αρχική Προτεραιότητα
Α	**0**	**10**	**0**
Β	**0**	**12**	**0**
Γ	**3**	**6**	**2**
Α	**4**	**1**	**3**

Οι διεργασίες αυτές εκτελούνται σε σύστημα με μη προεκτοπιστική (non-preemptive) χρονοδρομολόγηση πολυεπίπεδων, εκθετικών ουρών, με 4 κλάσεις προτεραιοτήτων και ισχυρότερη την κλάση προτεραιότητας 0. Το κβάντο της κλάσης 0 είναι 1 μονάδα χρόνου.
α) Δώστε το διάγραμμα Gannt της εκτέλεσης των παραπάνω εφαρμογών **σε** σύστημα με 1 επεξεργαστή. Επισημάνετε στο γραπτό τυχόν παραδοχές σας.
β) Υπολογίστε το μέσο χρόνο διεκπεραίωσης και το μέσο χρόνο αναμονής των διεργασιών στο σύστημα.

ΑΣΚΗΣΗ 2

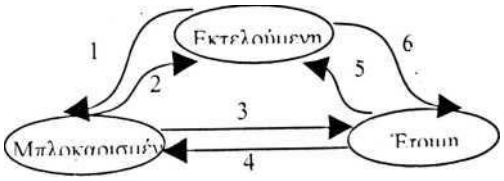
Τρεις διεργασίες Α, Β, και Γ, καταφθάνουν την χρονική στιγμή 0.0, 0,5 και 1.0 αντίστοιχα ενώ ο χρόνος εκτέλεσης (CPU burst time) είναι 8, 4, και 1 ms αντίστοιχα. Ποιος ο μέσος χρόνος διεκπεραίωσης για αυτές τις διεργασίες αν χρησιμοποιηθούν οι αλγόριθμοι (α) FCFS και (β) SJF; Σχεδιάστε τα χρονοδιαγράμματα Gantt που δείχνουν την εκτέλεση των διεργασιών για κάθε έναν από τους παραπάνω αλγόριθμους. Ποιος θα ήταν ο μέσος χρόνος διεκπεραίωσης για την περίπτωση του SJF, αν το σύστημα περίμενε για 1 μονάδα χρόνου πριν αρχίσει να εξυπηρετεί τις διεργασίες; Πως μπορεί ένα σύστημα χρονοπρογραμματισμού να «επιδιορθώσει» το γεγονός πως δεν μπορεί να προβλέψει το μέλλον;

ΑΣΚΗΣΗ 3

α) Σας δίνεται ο ακόλουθος κώδικας για τις διεργασίες Α και Β. Η κλήση work(i) εκτελεί υπολογισμούς, απασχολώντας τον επεξεργαστή για i μονάδες χρόνου. Η κλήση sleep(i) επιστρέφει τον επεξεργαστή στο σύστημα και μπλοκάρει τη διεργασία για i μονάδες πραγματικού χρόνου. Μετά το πέρας των i μονάδων χρόνου η διεργασία που κάλεσε τη sleep είναι και πάλι διαθέσιμη για εκτέλεση.

Α	**Β**
work(4);	work(2);
sleep(2);	sleep (4);
work(3);	work(7) ;
sleep (1);	sleep(1);
work(5);	work(l);
sleep(1);	
work(3);	

Οι διεργασίες Α και Β εκτελούνται σε ένα σύστημα που χρησιμοποιεί χρονοδρομολογητή πολυεπίπεδων εκθετικών ουρών με ανατροφοδότηση. Οι κλάσεις προτεραιοτήτων είναι 4, και το κβάντο στην κλάση 0 είναι 1 μονάδα χρόνου. Ο αλγόριθμος χρονοδρομολόγησης είναι μη προεκτοπιστικός στη διάρκεια κάθε κβάντου. Οι δύο διεργασίες έχουν αρχικό προτεραιότητα 0 και εισάγονται στο σύστημα τη χρονική στιγμή 0 (θεωρήστε ότι η Α φτάνει οριακά πριν τη Β). Δώστε το διάγραμμα Gantt της εκτέλεσης των διεργασιών στα σύστημα. Συνίσταται έντονα κάτω από το διάγραμμα Gantt να σχεδιάσετε τις ουρές των 4 κλάσεων προτεραιότητας και τις διεργασίες που υπάρχουν σε καθεμιά από τις ουρές κάθε χρονική στιγμή.
β) Δίπλα βλέπετε ένα απλοποιημένο διάγραμμα καταστάσεων διεργασίας. Για κάθε αριθμημένο βέλος, γράψτε τις συνθήκες που προκαλούν τη μετάβαση μεταξύ των καταστάσεων. Εάν για κάποιο αριθμημένο βέλος η μετάβαση δεν είναι δυνατή, πρέπει να το επισημάνετε.



γ) Περιγράψτε συνοπτικά, ή αν προτιμάτε σχηματικά, τη διαδικασία εναλλαγής περιβάλλοντος λειτουργίας (context switch) σε έναν επεξεργαστή.

ΑΣΚΗΣΗ 4

Υποθέστε ότι υπάρχουν τέσσερις διεργασίες PI, P2, P3, P4. με χρόνους άφιξης 0, 10, 20, 40, προτεραιότητες 10, 20, 30, 40 και χρόνους εκτέλεσης 30, 20, 50, 20. Οι διεργασίες αυτές χρονοπρογραμματίζονται μέσω μιας ουράς έτοιμων διεργασιών, σε μια μηχανή με έναν επεξεργαστή.
α) Ποιος είναι ο μέσος χρόνος ολοκλήρωσης της εκτέλεσης των παραπάνω διεργασιών με τους εξής αλγορίθμους χρονοπρογραμματισμού: (1) με προτεραιότητες (η υψηλότερη είναι αυτή με τη μικρότερη τιμή), (2) round-robin (ανά 20 μονάδες χρόνου), (3) η μικρότερη διεργασία πρώτα (SJF); Υποθέστε ότι το κόστος εναλλαγής περιβάλλοντος λειτουργίας είναι 1 μονάδα χρόνου, και ότι η άφιξη μιας διεργασίας οδηγεί σε διακοπή που ενεργοποιεί (εκ νέου) τον χρονοπρογραμματισμό.
β) Δίνεται ο παρακάτω ψευδοκώδικας χρονοπρογραμματισμού διεργασιών με προτεραιότητες και διακοπές:

```
create process(p) {
        …
        p->counter = p->priority;
        add_to_front_runqueue (p);
        …
}

schedule() {
        …
        c = 0;
        for (p = front_runqueue; p <= end_runqueue; p++) {
                if (p->counter > c) {
                        c = p->counter;
                        next = p;
                }
        }
        if (!c) {
                for (all processes p) {
                        p->counter = p->priority;
                }
        }
        run (next);
        …
}

do_timer_interrupt() {
        …
        next->counter--;
        if (next->counter < 0)
                schedule();
        …
}
```

Υποθέστε ότι κατά την διάρκεια της εκτέλεσης, η do_timer_interrupt καλείται σε κάθε διακοπή του χρονομετρητή, ανά 10 μονάδες χρόνου, για να ελέγξει το κατά πόσο πρέπει να επιλεγεί μια άλλη διεργασία.
(1) Σχεδιάστε το διάγραμμα Gantt που προκύπτει αν οι διεργασίες χρονοπρογραμματιστούν με αυτή τη μέθοδο, και με βάση τις προτεραιότητες που δόθηκαν παραπάνω.
(2) Ποιος ο μέσος χρόνος αναμονής για τις διεργασίες: Υποθέστε μηδενικό κόστος εναλλαγής περιβάλλοντος λειτουργίας. Υποθέστε επίσης ότι δεν υπάρχουν συνθήκες ανταγωνισμού στον κρίσιμο κώδικα της do_timer_interrupt.

ΑΣΚΗΣΗ 5

Παρακάτω σας δίνεται ο ψευδοκώδικας αλγορίθμου χρονοδρομολόγησης ενός συστήματος με πολλαπλούς επεξεργαστές. Η συνάρτηση new_process() καλείται οποτεδήποτε δημιουργείται μια νέα διεργασία. Η συνάρτηση clock_tick() καλείται από κάθε επεξεργαστή σε κάθε χτύπο του ρολογιού του συστήματος. Η rdy_q είναι ουρά που περιλαμβάνει τις έτοιμες προς εκτέλεση διεργασίες. Η running_process[] περιλαμβάνει δείκτες προς την εκτελούμενη διεργασία σε κάθε επεξεργαστή. Η idle_task είναι η διεργασία idle του συστήματος, η οποία δεν περιλαμβάνεται στην rdy_q. Η συνάρτηση proc_id() επιστρέφει τον αύξοντα αριθμό του επεξεργαστή που την εκτελεί. Τέλος, η χρήση των συναρτήσεων lock() / unlock() εξασφαλίζει αμοιβαίο αποκλεισμό μεταξύ των πολλαπλών επεξεργαστών στα σημεία που απαιτείται.

α) Περιγράψτε συνοπτικά τη βασική ιδέα του αλγορίθμου. Προσοχή! Αποφύγετε να εξηγήσετε τον κώδικα γραμμή προς γραμμή.

β) Ο αλγόριθμος είναι προεκτοπιστικός ή μη προεκτοπιστικός; Γιατί;

γ) Δώστε το διάγραμμα Gantt για την εκτέλεση των παρακάτω διεργασιών σε σύστημα με δύο επεξεργαστές. Αναφέρετε ξεκάθαρα οτιδήποτε παραδοχές τυχόν χρειαστεί να κάνετε. Θεωρήστε ότι ο χρόνος μεταγωγής περιβάλλοντος καθώς και η επιβάρυνση εκτέλεσης του χρονοδρομολογητή είναι αμελητέοι. Επίσης, θεωρείστε ότι οι διεργασίες είναι αμιγώς υπολογιστικές (δεν εκτελούν καθόλου I/O).

	Χρόνος Άφιξης	Χρόνος Εκτέλεσης	Στατική Προτερ.
A	0	3	2
B	0,5	4	3
C	1,5	3	2
D	2,5	2	1

δ) Υπολογίστε το μέσο χρόνο διεκπεραίωσης, καθώς και το μέσο χρόνο αναμονής των διεργασιών στο σύστημα.

ε) Με ποιο τρόπο μπορεί ο χρονοδρομολογητής να συμβάλει στην αποφυγή του thrashing σε ένα σύστημα όπου οι πραγματικές απαιτήσεις των διεργασιών σε μνήμη υπερβαίνουν τη διαθέσιμη φυσική μνήμη;

```
new_process(int static_prio) {
        …
        new_process->dynamic_prio = new process->static_prio = static_prio;
        lock();
        enqueue_rdy_q_end(new_process);
        unlock();
}

clock_tick() {
        int id = proc_id();
        if (running_process[id] 1= idle_task) {
                running_process[id]->dynamic_prio--;
        }
        schedule_this_proc();
}
```

```
schedule_this_proc() {
```

```
        int winner_prio = 0, id = proc_id ();
        lock();
        if (running_process[id] != idle_task)
                enqueue_rdy_q_front(running_process[id]);

        if (rdy_q_not_empty()) {
select:
                for each process p in rdy_q
                        if (p->dynamic_prio > winner_prio) {
                                winner_prio = p->dynamicprio;
                                running_proces s[id] = p;
                        }
                if (winner_prio == 0) {
                        for each process p in rdy_q
                                p->dynamic_prio = p->static_prio;
                                goto select;
                }
                dequeue_frora_rdy_q(running_process[id]);
        }
        else
                running__process[id] = idle_task;
        unlock();
        switch_this_proc to(running_proces s[id]);
}
```

ΑΣΚΗΣΗ 6

α) Αναφέρετε έναν αλγόριθμο χρονοδρομολόγησης του επεξεργαστή που εγγυάται τη μη ύπαρξη λιμοκτονίας (δεδομένου ενός πεπερασμένου αριθμού διεργασιών).
β) Είναι δυνατό οι επιλογές του αλγόριθμου χρονοδρομολόγησης να οδηγήσουν σε αδιέξοδο; Αν όχι, γιατί; Αν ναι, αναφέρετε ένα σύντομο παράδειγμα.
γ) Οι χρόνοι άφιξης (ΧΑ) και εκτέλεσης (ΧΕ) των διεργασιών Α, Β, Γ και Δ δίνονται στον πίνακα. Οι διεργασίες εκτελούνται σε σύστημα ενός επεξεργαστή, με χρονοδρομολογητή round-robin. Ο αλγόριθμος είναι μη προεκτοπιστικός στη διάρκεια του κβάντου. Το κβάντο είναι 2 μονάδες χρόνου, ενώ οι νέες διεργασίες στο σύστημα τίθενται στην αρχή της ουράς έτοιμων διεργασιών. Οι διεργασίες Α και Γ χρειάζονται το ίδιο lock καθ' όλη τη διάρκεια της εκτέλεσής τους. Η υλοποίηση του lock είναι τέτοια που μια διεργασία που βρίσκει το lock δεσμευμένο απελευθερώνει άμεσα τον επεξεργαστή της χωρίς όμως να μπλοκάρει, παραμένοντας δηλαδή στην ουρά έτοιμων διεργασιών. Θα διεκδικήσει εκ νέου το lock όταν επιλεγεί και πάλι από το χρονοδρομολογητή του επεξεργαστή. Δώστε το διάγραμμα Gantt της εκτέλεσης των τεσσάρων διεργασιών στο εν λόγω σύστημα.

	ΧΑ	ΧΕ
Α	0	7
Β	1	4
Γ	3	13
Δ	5	5

ΑΣΚΗΣΗ 7

Ο παρακάτω πίνακας σας δίνει τις απαιτήσεις σε χρόνο υπολογισμού, το χρόνο άφιξης και την αρχική προτεραιότητα 4 διεργασιών. Οι διεργασίες Α και Γ χρειάζεται καθ' όλη την εκτέλεσή τους να κατέχουν έναν (τον ίδιο) σημαφόρο, ο οποίος έχει αρχικοποιηθεί σε 1. Ο χρονοδρομολογητής του συστήματος βασίζεται στον αλγόριθμο εκθετικών ουρών με ανατροφοδότηση. Το σύστημα περιλαμβάνει 4 ουρές (0 έως 3). Ο χρονοδρομολογητής είναι μη προεκτοπιστικός στη διάρκεια κάθε κβάντου, ενώ το κβάντο για την ουρά προτεραιότητας 0 (την ισχυρότερη) είναι 1 msec. Ο χρόνος μεταγωγής περιβάλλοντος είναι αμελητέος. Τέλος, αμελητέος είναι ο χρόνος που απαιτείται για την — επιτυχημένη ή μη — προσπάθεια δέσμευσης του σημαφόρου. Να υπολογίσετε:

α) Το διάγραμμα Gantt της εκτέλεσης των διεργασιών σε ένα σύστημα με 1 επεξεργαστή.

β) Το μέσο χρόνο διεκπεραίωσης των διεργασιών

γ) Το μέσο χρόνο αναμονής των διεργασιών στο σύστημα

	Χρόνος Υπολογισμού	Χρόνος Άφιξης	Προτεραιότητα
Α	10	0	0
Β	4	1	2
Γ	7	2	1
Δ	6	4	2

SYNCHRONIZATION

ΑΣΚΗΣΗ 1

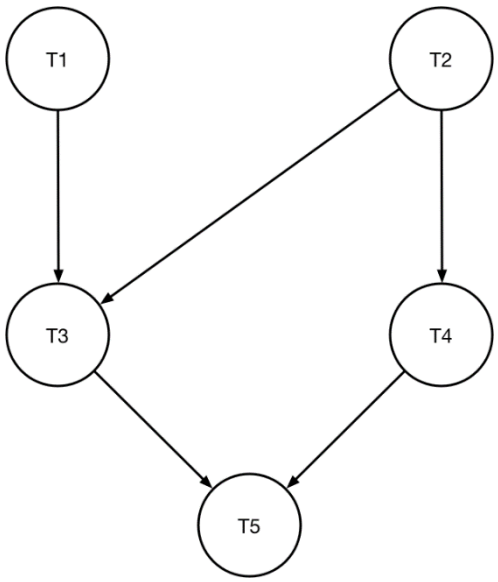
Ένα σύστημα έχει τέσσερις διεργασίες και πέντε πόρους. Η τρέχουσα ανάθεση και οι μέγιστες ανάγκες φαίνονται παρακάτω:

	Allocated	**Maximum**	**Available**
Process A	10211	11213	00x12
Process B	20110	22210	
Process C	11010	21310	
Process D	111101	11221	

Ποια η μικρότερη τιμή του x για την οποία αυτή είναι μια ασφαλής κατάσταση;

ΑΣΚΗΣΗ 2

α) Για το καλοκαιρινό μας πάρτυ έχουμε προσλάβει πολλούς bartenders που παρασκευάζουν δροσιστικά cocktails. Κάθε ποτήρι cocktail που ετοιμάζεται τοποθετείται σε έναν πάγκο ο οποίος χωράει n ποτήρια. Οι διψασμένοι καλεσμένοι μπορούν να πάνε στον πάγκο και να πάρουν ένα ποτήρι. Όταν οποιοσδήποτε από τους bartenders ετοιμάσει ένα cocktail και διαπιστώσει ότι ο πάγκος έχει γεμίσει, πρέπει να περιμένει να αδειάσει μία θέση (κάποιος καλεσμένος να πάρει ένα ποτό). Αντίστοιχα, αν ένας διψασμένος καλεσμένος βρει τον πάγκο άδειο, θα πρέπει να περιμένει έως ότου εμφανιστεί ένα cocktail. Επειδή τόσο οι bartenders όσο και οι καλεσμένοι ξέρουν από καλό συγχρονισμό... χρησιμοποιούν γενικούς (μετρητές) σηματοφόρους! Γράψτε τον ψευδοκώδικα που εκτελεί κάθε bartender και τον ψευδοκώδικα που εκτελεί κάθε καλεσμένος. Επίσης δώστε τις αρχικές τιμές του/των σηματοφόρου/ων που θα χρησιμοποιήσετε.
β) Δώστε τον ψευδοκώδικα των διεργασιών T1, T2, T3, T4, T5 ώστε με χρήση γενικών (μετρητών) σηματοφόρων να διασφαλίζεται η ακολουθία εκτέλεσης που περιγράφεται από τον παρακάτω γράφο. Συμβολίστε τον υπολογισμό κάθε διεργασίας Ti σαν compute(Ti). Προσπαθήστε να χρησιμοποιήσετε τους ελάχιστους δυνατούς σηματοφόρους. Μην ξεχάσετε να γράψετε τις αρχικές τους τιμές.



MEMORY

ΑΣΚΗΣΗ 1

Θεωρείστε τους ακόλουθους πίνακες τμημάτων (segment tables) για τις διεργασίες Α και Β:

# Τμήματος	Διεύθυνση Βάσης	Μέγεθος
0	700	200
1	400	200
2	1000	400

# Τμήματος	Διεύθυνση Βάσης	Μέγεθος
0	0	400
1	600	100
2	2000	300
3	1500	200

α) Αν μια λογική διεύθυνση αποτελείται από τέσσερα δεκαδικά ψηφία και το πρώτο ψηφίο είναι ο αριθμός του τμήματος ενώ τα τελευταία τρία η μετατόπιση, μεταφράστε τις ακόλουθες λογικές διευθύνσεις σε φυσικές, εφόσον αυτό είναι εφικτό. Αλλιώς αναφέρετε το είδος της μη έγκυρης πρόσβασης που δημιουργεί η φυσική διεύθυνση.

Διεργασία	Λογική Διεύθυνση
A	2350
A	1125
A	0021
B	0055
B	1036
B	4005
B	3305

β) Η προσέγγιση της κατάτμησης της μνήμης μπορεί να δημιουργήσει εξωτερικό κατακερματισμό Αν, ναι εξηγείστε σε ποια περίπτωση και δώστε μια λύση για το πρόβλημα του εξωτερικοί κατακερματισμού στην κατάτμηση της μνήμης.

ΑΣΚΗΣΗ 2

Ένας υπολογιστής με διευθύνσεις 32 bit χρησιμοποιεί έναν πίνακα σελίδων 2 επιπέδων. Οι εικονικές διευθύνσεις χωρίζονται σε ένα πεδίο 9 bits για τον πίνακα σελίδων πρώτου επιπέδου, σε ένα πεδίο 11 bits για τον πίνακα σελίδων δευτέρου επιπέδου, ενώ τα υπόλοιπα bits υποδεικνύουν την μετατόπιση μέσα στην σελίδα. Πόσο μεγάλες είναι οι σελίδες και πόσες υπάρχουν σε κάθε εικονικό χώρο διευθύνσεων; Γιατί και σε ποιες περιπτώσεις χρησιμοποιούνται πίνακες σελίδων πολλαπλών επιπέδων;

ΑΣΚΗΣΗ 3

Έστω ένα σύστημα με διευθύνσεις 32 bit, φυσική μνήμη 256Mbyte και πλαίσια 4Kbyte, και έστω πως κάθε διεργασία στο σύστημα πρέπει να έχει εικονική μνήμη 1Gbyte. Ορίστε ένα λογικό σχήμα σελιδοποίησης διπλού επιπέδου, δίνοντας την δομή των εικονικών διευθύνσεων. Επίσης, δώστε το επιπλέον κόστος σε μνήμη ανά διεργασία που συνεπάγεται το σχήμα σας για το σύστημα, αν υποθέσετε πως για να τρέξει μια διεργασία ο πίνακας πρώτου επιπέδου και τουλάχιστον ένας πίνακας δευτέρου επιπέδου πρέπει να βρίσκονται στην κυρίως μνήμη. Ποιο θα ήταν αυτό το κόστος αν χρησιμοποιούσατε σελιδοποίηση ενός επιπέδου;

ΑΣΚΗΣΗ 4

Θεωρείστε τον ακόλουθο πίνακα δύο διαστάσεων:

```
int count[100][100];
```

όπου το count[0][0] βρίσκεται στη θέση μνήμης 200, σε ένα σύστημα με σελιδοποίηση και σελίδες μεγέθους 200. Μια μικρή διεργασία είναι στη σελίδα 0 (θέσεις από 0 έως 199) για το χειρισμό του πίνακα (συνεπώς κάθε ανάκτηση εντολής θα γίνεται από τη σελίδα 0).
Για τρία πλαίσια σελίδας στη φυσική μνήμη, πόσα σφάλματα σελίδας δημιουργούνται από τους ακόλουθους βρόγχους αρχικοποίησης,- χρησιμοποιώντας αλγόριθμο αντικατάστασης σελίδων LRU, υποθέτοντας ότι στο ένα πλαίσιο είναι τοποθετημένη η διεργασία και τα δύο άλλα άδεια και ότι ένας πίνακας αποθηκεύεται στη μνήμη κατά γραμμή;

α)

```
for (j=0; j <100; j++)
        for (i=0; i<100; i++)
                count[i][j]=0;
```

β)

```
for (i=0; i<100; i++ )
        for (j = 0; j<l00; j++)
                count[i][j] =0;
```

ΑΣΚΗΣΗ 5

Θεωρείστε ένα σύστημα με 64MB φυσικής μνήμης, 32bit φυσικές διευθύνσεις, 32bit εικονικές διευθύνσεις και 4KB πλαίσια φυσικής μνήμης.
α) Χρησιμοποιώντας σχήμα σελιδοποίησης ενός επιπέδου, ποιος είναι ο μέγιστος αριθμός εγγραφών του πίνακα σελίδων για αυτό το σύστημα;
β) Χρησιμοποιώντας σχήμα σελιδοποίησης δύο επιπέδων με εξωτερικό πίνακα σελίδων 1024 εγγραφών, ποια θα ήταν η μετατόπιση σε εγγραφές σελίδας στον πίνακα που προσπελαύνεται για την εικονική διεύθυνση 0011 0000 0001 0000 1110 0010 0001 1100;

ΑΣΚΗΣΗ 6

Έστω σύστημα το οποίο υποστηρίζει σελιδοποίηση με μέγεθος ιδεατής σελίδας 4KB και ιδεατές διευθύνσεις των 33 bits. Ο μηχανισμός σελιδοποίησης υλοποιείται με τη βοήθεια πολυεπίπεδων πινάκων σελίδων. Ο πίνακας $1^{ου}$ επιπέδου χωράει σε 2 σελίδες, ενώ οι πίνακες των κατώτερων επιπέδων χωράνε καθένας σε 1 σελίδα. Κάθε εγγραφή, τέλος, στους πίνακες σελίδων, απαιτεί 4 bytes.

α) Πόσα επίπεδα πινάκων σελίδων χρησιμοποιούνται στο σύστημα;
β) Σε πόσες ομάδες bits χωρίζεται η ιδεατή διεύθυνση; Πόσα bits περιέχει η κάθε ομάδα και ποια είναι η «φυσική» της σημασία;
γ) Έστω η ιδεατή διεύθυνση 0 1011 0101 0100 0011 1010 0001 0011 1100. Πόσα bytes από την αρχή του πίνακα σελίδων $1^{ου}$ επιπέδου πρέπει να μετακινηθούμε, ώστε να βρούμε την πληροφορία που θα μας επιτρέψει να προχωρήσουμε στη διαδικασία μετάφρασης της ιδεατής σε φυσική διεύθυνση;
δ) Έστω ότι το κόστος των αναγνώσεων / εγγραφών μνήμης που οδηγούν σε σφάλμα σελίδας είναι 1 msec, ενώ το κόστος των εγγραφών / αναγνώσεων μνήμης που δεν οδηγούν σε σφάλμα σελίδας είναι 1 nsec. Θεωρήστε επίσης ότι έχετε κώδικα ο οποίος σε κάθε εντολή εκτελεί 2 προσπελάσεις στη μνήμη. Ο επεξεργαστής σας έχει ισχύ 10 MIPS και μπορείτε να τον διατηρείτε συνεχώς απασχολημένο, παρά τα σφάλματα σελίδων (π.χ. με χρήση πολυνηματισμού). Ποια θα πρέπει να είναι η συχνότητα των σφαλμάτων σελίδας (Page Fault Frequency - PFF σε σφάλματα σελίδας/sec) ώστε ο μέσος χρόνος προσπέλασης στη μνήμη να είναι μικρότερος από 50,00095 μsec; (Σημείωση: Τα τελικά αποτελέσματα είναι «στρογγυλοί» αριθμοί\ όσο και αν φαίνεται περίεργο στην πορεία. Αν δεν έχετε κομπιουτεράκι μπορείτε απλά να αφήσετε τα κλάσματα).
ε) Ποιοι είναι οι «αντικρουόμενοι» στόχοι που προσπαθούμε να εξισορροπήσουμε όταν επιλέγουμε το μέγεθος σελίδας ενός συστήματος;

ΑΣΚΗΣΗ 7

Έστω οι NxN πίνακες ακεραίων A και B (κάθε ακέραιος έχει μέγεθος 4 bytes) Καθένας από τους 2 πίνακες έχει μέγεθος 16 MB
α) Πόσο είναι το N;
β) Στους πίνακες εφαρμόζεται ο ακόλουθος κώδικας

```
for (i = 0; i < N; i++)
        for (j = 0; j < N; j++)
                A[i][j] = A[i][j] + B[i][j];

for (i = N-l; i >= 0; i--)
        for (j = 0; j <N; j++)
                A[i][j] = 2 * A[i][j];
```

Το σύστημα σας υποστηρίζει ιδεατές σελίδες των 8K. Τόσο ο πίνακας A όσο και ο B ξεκινούν από την αρχή κάποιας ιδεατής σελίδας (προφανώς διαφορετικής για τον καθένα) Ο κώδικας, με τη σειρά του, περιέχεται ολόκληρος σε μια ιδεατή σελίδα Το σύστημα παρέχει φυσική μνήμη μεγέθους 40K, η οποία αρχικά είναι άδεια. Πόσα σφάλματα σελίδων θα προκληθούν από την εκτέλεση του παραπάνω κώδικα εάν χρησιμοποιείται πολιτική αντικατάστασης σελίδων LRU.
γ) Έστω ότι το σύστημα υποστηρίζει ιδεατές διευθύνσεις των 58 bits. Έστω επίσης ότι κάθε εγγραφή στους πίνακες σελίδων απαιτεί 4 bytes. Πόσα επίπεδα πινάκων σελίδων απαιτούνται, εάν ο πίνακας σελίδων πρώτου επιπέδου πρέπει να χωράει σε 2 σελίδες ενώ οι πίνακες σελίδων των υπολοίπων επιπέδων πρέπει να χωράνε σε 1 σελίδα;
δ) Θεωρήστε ότι κάθε προσπέλαση στη μνήμη (σε δεδομένα ή σε πίνακες σελίδων) κοστίζει 1 μsec, Επίσης, κάθε επιτυχής προσπέλαση στον TLB κοστίζει 100 nsec. Οι ανεπιτυχείς προσπελάσεις στον TLB δε χρεώνονται (ολοκληρώνονται σε 0 nsec). Ποιο 0α πρέπει να είναι το ελάχιστο ποσοστό ευστοχίας στον TLB ώστε ο μέσος χρόνος προσπέλασης στη μνήμη να είναι κάτω από 1,29S μsec; Βασιστείτε στο ιεραρχικό σχήμα πινάκων σελίδων που υπολογίσατε στο (γ).

ΑΣΚΗΣΗ 8

Θεωρείστε ένα σύστημα με τρία φυσικά πλαίσια μνήμης στο οποίο δίνεται η παρακάτω ακολουθία αναφορών σελίδων μνήμης:
1, 3, 6, 7, 1, 3, 6, 7, 1, 3, 6, 7, 1, 3, 6, 7.
α) Ποιο είναι το πλήθος των σφαλμάτων σελίδας που θα συμβούν για κάθε έναν από τους ακόλουθους αλγόριθμους αντικατάστασης σελίδας:
(1) Βέλτιστο αλγόριθμο αντικατάστασης σελίδων
(2) LRU
(3) τον αλγόριθμο αντικατάστασης ρολογιού δεύτερης ευκαιρίας. Υπάρχει βέλτιστος αλγόριθμος αντικατάστασης σελίδων που δεν απαιτεί μελλοντική γνώση για κυκλικές ακολουθίες αναφορών όπως αυτή που φαίνεται παραπάνω; Αν ναι δώστε έναν γενικό αλγόριθμο. Αν όχι, εξηγήστε γιατί.

ΑΣΚΗΣΗ 9

α) Η ακόλουθη συλλογή δίσκων είναι οργανωμένη ως συστοιχία RAID-5, ενώ κάθε τομέας δίσκου περιέχει 1 byte. Ο δίσκος 1 έχει καταστραφεί. Επαναδημιουργήστε τα χαμένα δεδομένα.

	Δίσκος 0	Δίσκος 1	Δίσκος 2	Δίσκος 3
Τομέας 0	0x44		0x41	0x40
Τομέας 1	0x24		0x2D	0x4D
Τομέας 2	0x45		0x41	0x54

β) Εάν γνωρίζετε ότι οι γραμματοσκιασμένοι τομείς περιέχουν πλεονάζουσα πληροφορία ισοτιμίας μπορείτε – με βάση τον παρακάτω πίνακα που περιέχει την κωδικοποίηση χαρακτήρων ASCII να –

αποκρυπτογραφήσετε τα δεδομένα που υπήρχαν στη συστοιχία; Για παράδειγμα, ο χαρακτήρας ‘V’ αντιστοιχεί στον κωδικό ASCII 0x56

	0x0	**0x1**	**0x2**	**0x3**	**0x4**	**0x5**	**0x6**	**0x7**	**0x8**	**0x9**	**0xA**	**0xB**	**0xC**	**0xD**	**0xE**	**0xF**
0x0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0x1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0x2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0x3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0x4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0x5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
0x6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0x7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

γ) Ποιες δυνατότητες / πλεονεκτήματα προσφέρει η χρήση συστοιχίας RAID-0 σε σχέση με τη χρήση απλών δίσκων; Ποιο είναι το βασικό της μειονέκτημα και πως μπορεί να αντιμετωπιστεί;

ΑΣΚΗΣΗ 10

α) Είστε ο σχεδιαστής του λειτουργικού σε ένα σύστημα με ιδεατή διεύθυνση των 44 bits και σελίδες των 4KB. Πρέπει να αποφασίσετε αν σας συμφέρει να χρησιμοποιήσετε επίπεδο πίνακα σελίδων η ιεραρχικό 3 επίπεδων. Σε κάθε περίπτωση, οποιαδήποτε έγγραφη πίνακα σελίδων έχει εύρος 4 bytes. Επιπλέον, στην περίπτωση του ιεραρχικού πίνακα σελίδων, ο πίνακας σελίδων 1ου επιπέδου χωράει σε 4 σελίδες ενώ πίνακας σελίδων των χαμηλότερων επιπέδων χωράει σε 1 σελίδα Οι διεργασίες στο σύστημα σας αντίθετα με τα συνηθισμένα χρησιμοποιούν την ιδεατή μνήμη τους με συνεχή τρόπο Αν δηλαδή μια διεργασία έχει ενεργοποιήσει 4 MB ιδεατής μνήμης, αυτά θα αφορούν αποκλειστικά τις ιδεατές διευθύνσεις 0 έως $2^{22} - 1$. Ποια η επιβάρυνση σε μνήμη που προκαλούν οι πίνακες σελίδων κάθε διεργασίας για καθεμιά από τις δυο επιλογές, σε συνάρτηση με το μέγεθος του ενεργού χώρου ιδεατών διευθύνσεων της διεργασίας.
β) Εστία σύστημα με ιεραρχικό πίνακα σελίδων 2 επιπέδων. Το σύστημα προσφέρει 4 πλαίσια φυσικής μνήμης (αρχικά άδεια) και TLB 2 θέσεων (επίσης
αρχικά άδειο). Μια μετάφραση από ιδεατή σε φυσική διεύθυνση έχει μηδενικό κόστος, αν βρεθεί στον TLB. Σε αντίθετη περίπτωση, η μετάφραση γίνεται μέσω του ιεραρχικού πίνακα σελίδων. Κάθε προσπέλαση σε θέση φυσικής μνήμης που δεν προκαλεί σφάλμα σελίδας κοστίζει 10 μsec. Αν η προσπέλαση προκαλέσει σφάλμα σελίδας, το κόστος της είναι 10 msec. Για τη διευκόλυνσή σας, θεωρήστε ότι ο ιεραρχικός πίνακας σελίδων βρίσκεται προ-φορτωμένος σε φυσική μνήμη πέραν των 4 πλαισίων, η οποία μάλιστα είναι αρκετά μεγάλη ώστε ποτέ να μη συμβεί σφάλμα σελίδας κατά την προσπέλαση σε πίνακα σελίδων. Η πολίτικη αντικατάστασης είναι η LRU, τόσο για τα πλαίσια φυσικής μνήμης, όσο και για τις έγγραφες του TLB Σε πόσο χρόνο μπορεί να ολοκληρωθεί η ακολουθία προσπελάσεων στις παρακάτω ιδεατές διευθύνσεις,

0xdeadbeef, 0xcdacdacd, 0xbad2ceed, 0xcdacd123, 0xdeadb341, 0x34312315

ΑΣΚΗΣΗ 11

α) Ένα i-node περιέχει 12 δείκτες προς blocks δίσκου, εκ των οποίων 10 αξιοποιούνται ως άμεσες διευθύνσεις προς blocks, 1 ως έμμεση διεύθυνση ενός επιπέδου και ο τελευταίος ως έμμεση διεύθυνση επιπέδων. Κάθε διεύθυνση απαιτεί 4 bytes, ενώ κάθε block δίσκου έχει μέγεθος 1024 bytes. Ποιο είναι το μεγαλύτερο μέγεθος αρχείου που υποστηρίζεται από αυτό το σύστημα αρχείων;
β) Στον οδηγό δίσκου φτάνουν με ελάχιστη χρονική διαφορά μεταξύ τους, αλλά με αυτή τη σειρά, οι αιτήσεις για προσπελάσεις στους ακόλουθους κυλίνδρους 10, 30, 20, 22, 43, 12, 1, 28 Αρχικά οι κεφαλές βρίσκονται πάνω από τον κύλινδρο 25 και κινούνται προς τον κύλινδρο 0. Ο χρόνος αναζήτησης είναι 2 msec για μετακίνηση σε απόσταση ενός κυλίνδρου. Ποιος είναι ο συνολικός χρόνος αναζήτησης αν ο βραχίονας του δίσκου δρομολογείται με τον αλγόριθμο SSTF και ποιος αν η

δρομολόγηση γίνεται με τον αλγόριθμό SCAN;

ΑΣΚΗΣΗ 12

α) Τι είναι ο ελεγκτής DMA; Τι κερδίζουμε από τη χρήση του;
β) Η μονάδα εκτύπωσης ενός γρήγορου εκτυπωτή Laser έχει τεχνικά τη δυνατότητα να τυπώσει έως 64 σελίδες το λεπτό. Κάθε σελίδα περιέχει 64 γραμμές των 64 χαρακτήρων. Για την κωδικοποίηση κάθε χαρακτήρα απαιτούνται 2 bytes. Ο εκτυπωτής διαθέτει μνήμη 128 KB. Το ΛΣ αντιγράφει κάθε φορά ένα block χαρακτήρων, ικανό να γεμίσει τη μνήμη του εκτυπωτή, από τη μνήμη χρήστη στη μνήμη του ΛΣ. Σε δεύτερη φάση οι χαρακτήρες προωθούνται προς τη μνήμη του εκτυπωτή. Κάθε βήμα αντιγραφής από μνήμη σε μνήμη απαιτεί 1 μsec / byte. Όταν όλοι οι χαρακτήρες που βρίσκονται στη μνήμη του τυπωθούν, ενημερώνει το ΛΣ με μια διακοπή, η επεξεργασία της οποίας απαιτεί 100 msec. Τι ποσοστό χρόνου του επεξεργαστή δαπανάται για τις διαδικασίες I/O σε έναν υπολογιστή που ελέγχει τον εν λόγω εκτυπωτή, για την εκτύπωση μια πολύ μεγάλης εργασίας;

ΑΣΚΗΣΗ 13

Ένας σκληρός δίσκος περιστρέφεται με ταχύτητα 1200rpm. Στην επιφάνειά του υπάρχουν 64 τομείς ανά διαδρομή, με 512 bytes ανά τομέα. Ο δίσκος έχει 8 κεφαλές. Υποθέστε ότι μπορούν να γίνονται μεταφορές, και από τις 8 κεφαλές ταυτόχρονα. Ο δίσκος είναι τοποθετημένος σε ένα εξωτερικό κουτί που συνδέεται με τον υπολογιστή μέσω θύρας USB. Ο ιδιόκτητης σκέφτεται να αναβαθμίσει το εξωτερικό κουτί σε ένα που συνδέεται με θύρα eSATA
α) Θα υπάρξει διαφορά στην επίδοση (ρυθμό μεταφοράς δεδομένων); Σας δίνεται ότι ο ρυθμός μεταφοράς δεδομένων που υποστηρίζει ένας ελεγκτής USB και eSATA είναι 60 MBytes/sec και 300MBytes/sec αντίστοιχα. Δικαιολογήστε ποσοτικά την απάντηση σας
β) Σε κάθε μία από τις δύο περιπτώσεις, υπολογίστε το ποσοστό αξιοποίησης από το δίσκο του εύρους ζώνης που υποστηρίζει ο κάθε ελεγκτής;
γ) Στο τέλος κάθε αρχείου σε ένα δίσκο προσθέτουμε τη συμβολοσειρά “ABC" Είναι δυνατό ο ελεύθερος χώρος στο δίσκο να παραμείνει ο ίδιος; Γιατί,
δ) Ποιο πρόβλημα λύνουν τα συστήματα αρχείων με ημερολόγιο;

ΑΣΚΗΣΗ 14

α) Περιγράψτε συνοπτικά το μηχανισμό μεταγωγής περιβάλλοντος (context switch).
β) Γιατί είναι δύσκολη (αν και εφικτή) η υλοποίηση της μεταγωγής περιβάλλοντος με λογισμικό;
γ) Οι διεργασίες P1 και P2 εκτελούν αντίστοιχα τους κώδικες που φαίνονται παρακάτω. Η διεργασία PI εμφανίζεται στο σύστημα τη χρονική στιγμή 0, ενώ η P2 9μsec αργότερα. Το σύστημα περιλαμβάνει μνήμη 4 πλαισίων σελίδας των 4KB για δεδομένα, οργανωμένα σαν μια ιδιωτική δεξαμενή 2 πλαισίων για κάθε διεργασία, και η πολιτική αντικατάστασης που χρησιμοποιείται είναι η LRU. Αρχικά τα πλαίσια είναι άδεια. Υποθέστε ότι ο πίνακας Α κάθε διεργασίας ξεκινάει από την αρχή σελίδας, ενώ ο κώδικας και οι στοίβες των διεργασιών PI και P2 βρίσκονται μόνιμα σε κύρια μνήμη εκτός των 4 πλαισίων. Η συνάρτηση zero( ) μηδενίζει τη θέση του πίνακα Α που της δίνεται ως παράμετρος και απαιτεί 1μsec (μαζί με την εκτέλεση του κώδικα που αφορά τα for loops και την κλήση/επιστροφή της συνάρτησης zero()), ενώ η εξυπηρέτηση ενός σφάλματος σελίδας απαιτεί 4μsec. Η πολίτικη χρονοδρομολόγησης στο σύστημα είναι round-robin με κβάντο 5μsec. Ο χρόνος μεταγωγής περιβάλλοντος (context switch) είναι αμελητέος. Δώστε το διάγραμμα Gantt της εκτέλεση των δυο διεργασιών σε 1 επεξεργαστή.

P1	**P2**

int i, j;
struct {char c[1024];} A[2][8];
for (i = 0; i < 2; i++)
  for(j = 0; j < 8; j++)
  zero(&A[i][j]);
	int i, j;
	struct {char c[1024];} A[2][8];
	for(j = 0; j < 8; j++)
	  for (i = 0; i < 2; i++)
	  zero(&A[i][j]);