Τμημα ΜΗΥΤΔ, Πανεπιστημιο Θεσσαλιας
Λειτουργικα Συστηματα (ΗΥ222)
Διδασκων: Χρηστος Δ. Αντωνοπουλος
Τελική Εξέταση Σεπτεμβρίου 2009, Χρόνος: 2:30

Θέμα 1 (3 μονάδες)
α) (0,75 μονάδες) Περιγράψτε την τεχνική των ομαδοποιημένων λιστών ελευθέρων αντικειμένων για τη δυναμική διαχείριση μνήμης. Ποιο είναι το βασικό μειονέκτημά της;

Θεωρία, διάλεξη 15, σελ. 21.

β) (0,5 μονάδες) Είναι δυνατό να επιτευχθεί αυτόματη επιστροφή της δυναμικά δεσμευμένης μνήμης, όταν αυτή δε χρειάζεται πλέον; Ναι ή όχι και πώς / γιατί;

Θεωρία, διάλεξη 15, σελ. 25

γ) (1 μονάδες) Περιγράψτε τον αλγόριθμο "mark & sweep". Ποιο είναι το μεγάλο του πλεονέκτημα;

Θεωρία, διάλεξη 15, σελ. 28.

δ) (0,75 μονάδες) Είναι δυνατόν να υλοποιηθεί ο αλγόριθμος "mark & sweep" χωρίς υποστήριξη από το μεταγλωττιστή; Αν όχι, γιατί; Αν ναι, πώς;

Θεωρία, διάλεξη 15, σελ. 29

Θέμα 2 (3,5 μονάδες)
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) (1 μονάδα) Περιγράψτε συνοπτικά τη βασική ιδέα του αλγορίθμου. Προσοχή! Αποφύγετε να εξηγήσετε τον κώδικα γραμμή προς γραμμή.

Βασική δομή δεδομένων είναι η ουρά των έτοιμων προς εκτέλεση διεργασιών. Οι προσπελάσεις στην ουρά προστατεύονται από ζεύγη lock() / unlock() ώστε να διασφαλισθεί ότι δε θα γίνουν ακριβώς την ίδια στιγμή από παραπάνω από έναν επεξεργαστές. Αυτό θα μπορούσε να έχει ως συνέπεια να βρεθεί η δομή σε ασυνεπή κατάσταση, ή παραπάνω από 1 επεξεργαστές να επιλέξουν προς εκτέλεση την ίδια διεργασία στο ίδιο κβάντο.

Ο κώδικας των new_process(), clock_tick() και schedule_this_proc() εκτελείται ανεξάρτητα από κάθε επεξεργαστή και για λογαριασμό του.

Όταν δημιουργείται μια νέα διεργασία η στατική και η δυναμική προτεραιότητά της αρχικοποιούνται στην ίδια τιμή (την τιμή που περνιέται ως παράμετρος στη συνάρτηση new_process()). Η νέα διεργασία τοποθετείται στο τέλος της ουράς των έτοιμων διεργασιών. Προσέξτε ότι δεν καλείται η συνάρτηση χρονοδρομολόγησης (άρα δεν μπορεί κάποιος επεξεργαστής να ξεκινάει την εκτέλεση μιας νέας διεργασίας, που μόλις ήρθε, στο μέσο μεταξύ δύο διακριτών χρονικών στιγμών).

Σε κάθε χτύπο ρολογιού μειώνεται κατά ένα η δυναμική προτεραιότητα της διεργασίας που έτρεχε στον κάθε επεξεργαστή (η συνάρτηση εκτελείται ξεχωριστά από κάθε επεξεργαστή). Εξαίρεση είναι η περίπτωση που η εκτελούμενη διεργασία ήταν το idle task. Ακολούθως, εκτελείται ο αλγόριθμος χρονοδρομολόγησης.

Η βασική ιδέα του αλγόριθμου χρονοδρομολόγησης είναι ότι από της διαθέσιμες προς εκτέλεση διεργασίες επιλέγεται κάθε φορά αυτή με τη μεγαλύτερη δυναμική προτεραιότητα. Σε περίπτωση που οι δυναμικές προτεραιότητες όλων των διεργασιών στην ουρά έχουν γίνει 0, αυτές επαναρχικοποιούνται στις στατικές τους προτεραιότητες. Η διεργασία που επελέγη προς εκτέλεση αφαιρείται από την ουρά. Θα επανατοποθετηθεί στην ουρά (και μάλιστα στην αρχή της) κατά την επόμενη εκτέλεση του χρονοδρομολογητή για τον συγκεκριμένο επεξεργαστή. Σε περίπτωση που η ουρά των έτοιμων προς εκτέλεση διεργασιών είναι άδεια, εκτελείται το idle task. Προσέξτε ότι: α) Όταν επανα-αρχικοποιηθούν οι προτεραιότητες, δεν αρχικοποιούνται οι προτεραιότητες τυχόν διεργασιών που εκτελούνται, αφού αυτές βρίσκονται εκτός της ουράς και β) Η διεργασία που έτρεχε σε κάποιον επεξεργαστή θα προτιμηθεί και για το επόμενο κβάντο μεταξύ τυχόν άλλων διεργασιών με την ίδια προτεραιότητα. Αυτό φυσικά δεν ισχύει αν υπάρχει στην ουρά διεργασία με μεγαλύτερη προτεραιότητα.

β) (0,25 μονάδες) Ο αλγόριθμος είναι προεκτοπιστικός ή μη προεκτοπιστικός; Γιατί;

Ο αλγόριθμος είναι προεκτοπιστικός. Οι διεργασίες δεν αφήνονται να τελειώσουν την εκτέλεσή τους. Είναι πιθανόν να τους αφαιρεθεί ο επεξεργαστής από το λειτουργικό. Ο κώδικας χρονοδρομολόγησης καλείται σε κάθε χτύπο του ρολογιού.

γ) (1,25 μονάδες) 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

Στον πίνακα φαίνονται στον εκθέτη κάθε διεργασίας ο εναπομένων χρόνος εκτέλεσής της και στον δείκτη η εναπομένουσα δυναμική προτεραιότητα στο τέλος κάθε χρονικής μονάδας, πριν επανατοποθετηθούν στην ουρά οι διεργασίες που εκτελούνται στους επεξεργαστές. Στη σειρά Q απεικονίζεται η κατάσταση της ουράς στο τέλος κάθε χρονικής μονάδας. Θεωρούμε ότι ο επεξεργαστής P1 διαλέγει πάντα ελάχιστα νωρίτερα από τον P2 διεργασία προς εκτέλεση.

Προσέξτε ότι τη χρονική στιγμή 2 η διεργασία B θα παραμείνει στον επεξεργαστή P1. Το ίδιο θα γίνει τη χρονική στιγμή 3 (οι B και C δε θα παραχωρήσουν τον επεξεργαστή στην D). Τη χρονική στιγμή 4, επιλέγεται κατ' αρχήν η D στον επεξεργαστή P1. Ο P2 βρίσκει όλες τις διεργασίες με δυναμική προτεραιότητα 0 και ανανεώνει τις τιμές τους. Προσέξτε ότι η D είναι εκτός ουράς και άρα η προτεραιότητά της δε θα ανανεωθεί. Ακολούθως ο P2 θα επιλέξει τη B ως έχουσα τη μεγαλύτερη δυναμική προτεραιότητα. Τέλος, τη χρονική στιγμή 6 ανανεώνεται η δυναμική προτεραιότητα της D.

	0	*1*	*2*	*3*	*4*	*5*	*6*	*7*
P1		A^2_1	B^3_2	B^2_1	B^1_0	D^1_0	C^0_1	D^0_0
P2			A^1_0	C^2_1	C^1_0	B^0_2	A^0_1	
Q	A^3_2	B^4_3	C^3_2	A^1_0 D^2_1	A^1_0 D^2_1	C^1_2 A^1_2	D^1_0	

δ) (0,5 μονάδες) Υπολογίστε το μέσο χρόνο διεκπεραίωσης, καθώς και το μέσο χρόνο αναμονής των διεργασιών στο σύστημα.

Για κάθε διεργασία ο χρόνος διεκπεραίωσης (ΧΔ) υπολογίζεται ως χρόνος τερματισμού (ΧΤ) – χρόνος άφιξης (ΧΑ). Άρα ΜΧΔ = [(6-0) + (5-0,5) + (6-1,5) + (7-2,5)]/4 = 4,875

Για κάθε διεργασία ο χρόνος αναμονης (ΧΑ) υπολογίζεται ως ο χρόνος διεκπεραίωσης (ΧΔ) – χρόνος εκτέλεσης (ΧΕ). Άρα ΜΧΑ = [(6-3) + (4,5-4) + (4,5-3) + (4,5-2)]/4 = 1.875

ε) (0,5 μονάδες) 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;

Θεωρία, διάλεξη 8, σελ. 19, 20

```
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] != 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->dynamic_prio;
				running_process[id] = p;
			}
		if (winner_prio == 0) {
			for each process p in rdy_q
				p->dynamic_prio = p->static_prio;
			goto select;
		}
		dequeue_from_rdy_q(running_process[id]);
	}
	else
		running_process[id] = idle_task;
	unlock();
	switch_this_proc_to(running_process[id]);
}
```

<u>Θέμα 3</u> (3,5 μονάδες)
Υποθέστε ότι σε ένα δίσκο ο χρόνος αναζήτησης είναι της μορφής t = x + yL, όπου t είναι ο χρόνος (σε msec) και L η απόσταση μετακίνησης της κεφαλής (σε κυλίνδρους). Υποθέστε ότι ο χρόνος αναζήτησης για μετακίνηση στον αμέσως γειτονικό κύλινδρο είναι 1 msec και είναι 0.5 επιπλέον msec για κάθε επιπλέον κύλινδρο. Ο δίσκος έχει 7000 κυλίνδρους (0 έως 6999). Η κεφαλή επί του παρόντος εξυπηρετεί μια αίτηση στον κύλινδρο 143, ενώ η προηγούμενη αίτηση που εξυπηρετήθηκε βρισκόταν στον κύλινδρο 125. Η ουρά των προς εξυπηρέτηση αιτήσεων είναι (με σειρά άφιξης): 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130.
α) (0,25 μονάδες) Υπολογίστε τους συντελεστές x και y της παραπάνω εξίσωσης.

Αφού ο χρόνος για τη μετακίνηση κατά 1 μόνο κύλινδρο είναι 1msec, ισχύει 1 = x+y. Αφού ο χρόνος για την μετακίνηση για έναν επιπλέον κύλινδρο είναι 0,5msec επιπλέον, ισχύει 1,5 = x+2y. Από τις 2 αυτές σχέσεις προκύπτει εύκολα ότι x = y = 0,5.

β) (1,25 μονάδες) Υπολογίστε το συνολικό χρόνο αναζήτησης για την εξυπηρέτηση των παραπάνω αιτήσεων όταν χρησιμοποιείται αλγόριθμος FCFS και αλγόριθμος LOOK.

Cyl_{prev} και Cyl_{curr} είναι η προηγούμενη και η επόμενη θέση της κεφαλής αντίστοιχα. L η απόσταση μεταξύ των 2 θέσεων και t ο χρόνος που απαιτείται για τη μετακίνηση (σύμφωνα είτε με τον τύπο του (α) είτε με τα δεδομένα της εκφώνησης). Για την περίπτωση του LOOK, προσέξτε ότι αφού η προηγούμενη θέση της κεφαλής ήταν η 125 και η τρέχουσα η 143, η κεφαλή κινείται προς τους μεγαλύτερους κυλίνδρους. Συνεπώς οι αιτήσεις στους κυλίνδρους 130 και 86 θα εξυπηρετηθούν στο τέλος, όταν η κεφαλή γυρίσει από τους μεγαλύτερους προς τους μικρότερους κυλίνδρους, και μάλιστα με αυτή τη σειρά. Επίσης, προσέξτε ότι στον αλγόριθμο LOOK η κεφαλή δε φτάνει στα άκρα του δίσκου.

FCFS			
Cyl_{prev}	***Cyl_{curr}***	***L***	***t***
143	*86*	*57*	*29*
86	*1470*	*1384*	*692.5*
1470	*913*	*557*	*279*
913	*1774*	*861*	*431*
1774	*948*	*826*	*413.5*
948	*1509*	*561*	*281*
1509	*1022*	*487*	*244*
1022	*1750*	*728*	*364.5*
1750	*130*	*1620*	*810.5*
		Σύνολο	*3545*

LOOK			
Cyl_{prev}	***Cyl_{curr}***	***L***	***t***
143	*913*	*770*	*385.5*
913	*948*	*35*	*18*
948	*1022*	*74*	*37.5*
1022	*1470*	*448*	*224.5*
1470	*1509*	*39*	*20*
1509	*1750*	*241*	*121*
1750	*1774*	*24*	*12.5*
1774	*130*	*1644*	*822.5*
130	*86*	*44*	*22.5*
		Σύνολο	*1664*

γ) (0,5 μονάδες) Υποθέστε ότι ο δίσκος περιστρέφεται με 5000 rpm. Τι απόσταση (πόσους κυλίνδρους) μπορεί να καλύψει η κεφαλή στο χρονικό διάστημα που απαιτείται για να ολοκληρωθεί 1 περιστροφή;

5000rpm => 5000/60 revolutions per sec => 12 msec / revolution.
Αντικαθιστώντας στον τύπο του (α) και λύνοντας ως προς L, προκύπτει ότι η κεφαλή μετακινείται κατά 23 κυλίνδρους.

δ) (0,5 μονάδα) Ο παραπάνω δίσκος έχει τομείς (sectors) των 512 bytes, 160 τομείς ανά διαδρομή (sectors/track), και 20 κεφαλές (δηλαδή 10 πλάκες, με 2 μαγνητικές επιφάνειες σε κάθε πλάκα). Μόνο μία κεφαλή μπορεί να είναι ενεργή κάθε στιγμή. Ο χρόνος για να ενεργοποιήσουμε διαφορετική κεφαλή είναι

αμελητέος. Ποιός είναι ο μέγιστος (έστω και για μικρά διαστήματα) ρυθμός μεταφοράς δεδομένων που μπορεί να επιτευχθεί από αυτό το δίσκο;

Ο μέγιστος ρυθμός μεταφοράς δεδομένων μπορεί να επιτευχθεί μόνο όσο η κεφαλή δε μετακινείται. Κατά τη διάρκεια μιας περιστροφής του δίσκου, και με την κεφαλή πάνω από ένα συγκεκριμένο κύλινδρο, μπορούν να μεταφερθούν:
*512 Bytes / sector * 160 sectors/track * 1 track = 81920 Bytes = 80KB*
Ο χρόνος περιστροφής είναι 12msec (από (γ)). Άρα μεταφέρονται 81920 Bytes/12msec ή 6.826.666,67 Bytes/sec.

ε) (1 μονάδα) Ποιος είναι ο μέσος ρυθμός μεταφοράς δεδομένων που μπορεί να επιτευχθεί για μια πολύ μεγάλη μεταφορά;

Ας υποθέσουμε ότι θέλουμε να μεταφέρουμε τα δεδομένα όλου του δίσκου.
Από προηγουμένως βρήκαμε ότι κάθε διαδρομή αποθηκεύει 81.920 Bytes. Σε κάθε επιφάνεια υπάρχουν 7.000 διαδρομές, άρα 573.440.000 Bytes. Συνολικά οι επιφάνειες είναι 20, άρα ο δίσκος αποθηκεύει 11.468.800.000 Bytes.

*Για να διαβαστούν αυτά τα δεδομένα, θα πρέπει να ολοκληρωθεί 1 περιστροφή για κάθε διαδρομή. Οι διαδρομές συνολικά είναι 7.000 * 20 = 140.000, άρα απαιτούνται 140.000 περιστροφές. Δεδομένης της ταχύτητας περιστροφής του δίσκου (5000 rpm) απαιτούνται συνεπώς 28 λεπτά, ή 1680sec. Επιπλέον, η κεφαλή θα πρέπει να σαρώσει όλους τους κυλίνδρους. Χωρίς να αναγκάζεται να πισογυρίσει ποτέ, θα αναγκαστεί να μετακινηθεί 6999 φορές σε απόσταση 1 κυλίνδρου. Αυτή η μετακίνηση απαιτεί περίπου 7sec. Συνολικά λοιπόν θα χρειαστούν 1687 sec.*

Συνοψίζοντας, σε 1687 sec μεταφέρονται 11.468.800.000 Bytes, ή διαφορετικά 6.798.340,25 Bytes/sec

<u>Θέμα 4</u> (2 μονάδες)
α) (0,5 μονάδες) Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα της σελιδοποίησης έναντι της τμηματοποίησης;

Θεωρία, διάλεξη 6, σελ. 2

β) (1,5 μονάδες) Έστω σύστημα με τους πίνακες σελίδων του πάντα αποθηκευμένους στην κύρια μνήμη. Το σύστημα έχει ιδεατή διεύθυνση των 64 bit και σελίδες των 4KB. Κάθε εγγραφή πίνακα σελίδων απαιτεί 4 bytes. Ο πίνακας σελίδων 1ου επιπέδου καταλαμβάνει 16KB, ενώ οι πίνακες σελίδων των υπολοίπων επιπέδων 4MB έκαστος. Μια προσπέλαση στη φυσική μνήμη ολοκληρώνεται σε 200 nsec. Ποιος είναι ο μέσος χρόνος προσπέλασης στην ιδεατή μνήμη, αν το ποσοστό ευστοχίας στον TLB είναι 75%; Υποθέστε ότι η επιβάρυνση προσπέλασης στον TLB είναι 0, τόσο σε περίπτωση ευστοχίας, όσο και σε περίπτωση αστοχίας.

Εφόσον οι σελίδες είναι των 4KB (2^{12} Bytes) απαιτούνται 12 bits της ιδεατής διεύθυνσης για να εκφραστεί η μετατόπιση στο εσωτερικό της σελίδας. Συνεπώς απομένουν 52 bits της ιδεατής διεύθυνσης για τον αριθμό σελίδας.

Ο πίνακας σελίδων 1ου επιπέδου καταλαμβάνει 16KB (2^{14} Bytes) με 4 Bytes / θέση, συνεπώς περιέχει 2^{12} θέσεις. Άρα, τα 12 ποιο σημαντικά bits της ιδεατής διεύθυνσης αξιοποιούνται για να μας δώσουν τη θέση του πίνακα σελίδων 1ου επιπέδου που πρέπει να προσπελάσουμε ώστε να βρούμε την αρχή του κατάλληλου πίνακα σελίδων 2ου επιπέδου για να συνεχίσουμε τη μετάφραση.

Οι πίνακες σελίδων των υπολοίπων επιπέδων χωράνε σε 4MB (2^{22} Bytes) με 4 Bytes/θέση, συνεπώς περιέχουν 2^{20} θέσεις έκαστος. Παρόμοια με παραπάνω, χρειάζονται 20 bits της ιδεατής διεύθυνσης για καθένα από τα επιπλέον (του 1ου) επίπεδα πινάκων σελίδων.

Συνολικά καταλήγουμε ότι στο σύστημά μας χρησιμοποιείται πίνακας σελίδων 3 επιπέδων.

Το κόστος πράξεων στη μνήμη είναι το άθροισμα του κόστους μετάφρασης ιδεατής σε φυσική διεύθυνση συν το κόστος της προσπέλασης αυτής καθεαυτής (200 nsec).

Το κόστος μετάφρασης είναι 0 nsec στην περίπτωση ενός TLB hit και 600nsec στην περίπτωση TLB miss, αφού τότε θα χρειαστεί να διατρέξουμε τα 3 επίπεδα του πίνακα σελίδων, κάνοντας 1 προσπέλαση μνήμης για κάθε επίπεδο.

Αντικαθιστώντας στο γνωστό τύπο:
$MXΠ = h*TLB_{penalty} + (1-h)PageTable_{penalty} + AccessCost = 0.75*0\ nsec + 0.25*600\ nsec + 200\ nsec = 350\ nsec$