Λειτουργικά Συστήματα
Θέματα-Λύσεις
(Τμήμα Ρουμελιώτη)

ΠΕΡΙΕΧΟΜΕΝΑ:

- Σεπτέμβριος 2020 –με λύσεις
- Ιούνιος 2020 –με λύσεις
- Φεβρουάριος 2021 –με λύσεις
- Σεπτέμβριος 2021
- Σεπτέμβριος 2022
- Φεβρουάριος 2023 –με λύσεις
- Φεβρουάριος 2024 –με λύσεις

Σεπτέμβριος 2020

Θέμα 1ο (20 μονάδες)
Δίνεται η παρακάτω σειρά αιτήσεων για σελίδες της ιδεατής μνήμης: (το Χ είναι από τον ΑΜ σας)

X, 10, 12, X, 16, 10, X, 16, 13, 14, X, 15, 10, X, 14, 15, 21, 19, X, 21

Θεωρήστε ότι η φυσική μνήμη χωράει 8 πλαίσια και ότι το TLB μπορεί να αποθηκεύσει 4 σελίδες. Ο πίνακας χαρτογράφησης σελίδων (αποθηκευμένος στην μνήμη) είναι ο ακόλουθος:

Σελίδα ιδεατής μνήμης	Αντίστοιχο πλαίσιο φυσικής μνήμης
0	-
1	-
2	6
3	-
4	-
5	-
6	-
7	7
8	-
9	-
10	1

Σελίδα ιδεατής μνήμης	Αντίστοιχο πλαίσιο φυσικής μνήμης
11	4
12	5
13	-
14	-
15	-
16	0
17	-
18	-
19	-
20	2
21	3

Πριν ξεκινήσουν οι παραπάνω αιτήσεις, ο TLB έχει τα περιεχόμενα που δίνονται παρακάτω. Θεωρήστε ότι το TLB και η φυσική μνήμη χρησιμοποιούν πολιτική FIFO για την αντικατάσταση των σελίδων και οι σελίδες έχουν μπει με τη σειρά, ξεκινώντας από τη μικρότερη διεύθυνση της μνήμης και του TLB.

Σελίδα ιδεατής μνήμης	Πλαίσιο φυσικής μνήμης
16	0
10	1
20	2
21	3

1) Να δείξετε την τελική κατάσταση του TLB για τη δοθείσα σειρά αιτήσεων.
2) Ποιο είναι το hit ratio του TLB;
3) Πόσες αναφορές στην μνήμη θα γίνουν για την ανάγνωση των παραπάνω σελίδων, αγνοώντας την μεταφορά σελίδας κατά το page fault;
4) Να επαναλάβετε το ερώτημα (3) αν δεν υπήρχε TLB.

Λύση
Για X<>(2,7) τα σφάλματα σελίδας και σφάλματα TLB δίνονται στον παρακάτω πίνακα:

	X	10	12	X	16	10	X	16	13	14	X	15	10	X	14	15	21	19	X	21
PF	*	✓	✓	✓	*	*	✓	✓	*	*	✓	*	✓	✓	✓	✓	*	*	✓	✓
TLB	*	✓	*	✓	*	*	✓	✓	*	*	*	*	*	✓	✓	✓	*	*	*	✓

1. Η τελική κατάσταση του TLB είναι:

Page #	Frame #
10	**2**
21	**6**
19	**7**
X	**0**

2. Από τον πρώτο πίνακα το hit ratio του TLB είναι 8/20 = **40%**
3. Σε κάθε TLB hit μία ανάγνωση. Σε κάθε TLB miss, 2 αναγνώσεις (αφού θα διαβαστεί και ο PMT). Άρα συνολικά $8 \times 1 + 12 \times 2 = \mathbf{32}$.
4. Αν δεν υπήρχε ο TLB για κάθε ανάγνωση θα θέλαμε δύο αναγνώσεις, άρα συνολικά **40**.

Για X = 2, ή X=7 τα σφάλματα σελίδας και σφάλματα TLB δίνονται στον παρακάτω πίνακα (όταν το πλαίσιο που αλλάζει είναι στο TLB, τότε η νέα σελίδα πηγαίνει στην αντίστοιχη γραμμή του TLB ανεξαρτήτως σειράς FIFO):

	X	10	12	X	16	10	X	16	13	14	X	15	10	X	14	15	21	19	X	21
PF	✓	✓	✓	✓	✓	✓	✓	✓	*	*	✓	*	*	✓	✓	✓	*	*	✓	✓
TLB	*	✓	*	✓	*	*	✓	✓	*	*	✓	*	*	*	✓	✓	*	*	✓	✓

1. Η τελική κατάσταση του TLB είναι:

Page #	Frame #
19	**5**
10	**3**
X	**6 ή 7**
21	**4**

2. Από τον πρώτο πίνακα το hit ratio του TLB είναι 9/20 = **45%**
3. Σε κάθε TLB hit μία ανάγνωση. Σε κάθε TLB miss, 2 αναγνώσεις (αφού θα διαβαστεί και ο PMT). Άρα συνολικά $9 \times 1 + 11 \times 2 = \mathbf{31}$.
4. Αν δεν υπήρχε ο TLB για κάθε ανάγνωση θα θέλαμε δύο αναγνώσεις, άρα συνολικά **40**.

Σεπτέμβριος 2020

Θέμα 2ο (15 μονάδες)

Θεωρήστε ένα σύστημα, το οποίο εκτελεί 5 διεργασίες. Να ορίσετε τους σηματοφορείς (με τις αρχικές τιμές τους) και τον τρόπο χρησιμοποίησης των λειτουργιών up/down (χωρίς να αλλάξετε με άλλον τρόπο τον κώδικα ή να χρησιμοποιήσετε άλλες μεταβλητές), έτσι ώστε να είναι δυνατή η **επαναληπτική** εκτύπωση της συμβολοσειράς:

AABBCCDDDE

Διεργασία 1	Διεργασία 2	Διεργασία 3	Διεργασία 4	Διεργασία 5
....	....	....	....	....
Loop	Loop	Loop	Loop	Loop
....	....	....	....	....
Print A	Print B	Print C	Print D	Print E
.....	.....	.....	.....	.....
Until forever	Until forever	Until forever	Until forever	Until forever

Λύση

Semaphore Sa=2, Sb=0, Sc=0. Sd=0, Se=0

Διεργασία 1	Διεργασία 2	Διεργασία 3	Διεργασία 4	Διεργασία 5
....	....	....	....	....
Loop	Loop	Loop	Loop	Loop
....	....	....	....	....
Down (Sa)	Down (Sb)	Down (Sc)	Down (Sd)	Down (Se)
Print A	Print B	Print C	Print D	Down (Se)
Up(Se)	Up(Se)	Up(Se)	Up(Se)	Up(Sb)
.....	.....	.....	.....	Up(Sb)
Until forever	Until forever	Until forever	Until forever	Down (Se)
				Down (Se)
				Up(Sc)
				Up(Sc)
				Down (Se)
				Down (Se)
				Up(Sd)
				Up(Sd)
				Up(Sd)
				Down (Se)
				Down (Se)
				Down (Se)
				Print E
				Up(Sa)
				Up(Sa)
				.....
				Until forever

Προσέξτε ότι αν η διεργασία Ε είχε πάνω από ένα Print η επαναληπτική διαδικασία δεν θα ήταν με κανένα τρόπο εφικτή.

Σεπτέμβριος 2020

Θέμα 3ο (15 μονάδες)

Έστω ότι σε ένα σύστημα υπάρχουν μόνο δύο διεργασίες Δ0 και Δ1 που τρέχουν με σταθερά κβάντα 100 χρονικές μονάδες και χρησιμοποιούν τη λύση του Peterson, όπως φαίνεται παρακάτω. Έστω ότι ξεκινάει αρχικά η Δ1 και οι 100 μονάδες αρκούν μέχρι να τεθεί turn=process (αφού αλλάξει η turn). Για την Δ0, οι 100 μονάδες αρκούν για να μπει στο βρόχο while της εισαγωγής στο κρίσιμο τμήμα. Σε ποια χρονική στιγμή θα μπουν σε ΚΤ οι δύο διεργασίες (αν θα μπουν); Ξεκινήστε από το χρόνο t=0 και θεωρήστε τον χρόνο του ΚΤ καθώς και τον χρόνο μεμονωμένων εντολών ως αμελητέο.

```
#define FALSE  0
#define TRUE   1
#define N      2                        /* number of processes */

int turn;                               /* whose turn is it? */
int interested[N];                      /* all values initially 0 (FALSE) */

void enter_region(int process);         /* process is 0 or 1 */
{
     int other;                         /* number of the other process */

     other = 1 – process;               /* the opposite of process */
     interested[process] = TRUE;        /* show that you are interested */
     turn = process;                    /* set flag */
     while (turn == process && interested[other] == TRUE) /* null statement */ ;
}

void leave_region(int process)          /* process: who is leaving */
{
     interested[process] = FALSE;       /* indicate departure from critical region */
}
```

Λύση

Η σειρά εναλλαγής είναι Δ1-Δ2-Δ1(ΚΤ)-Δ2(ΚΤ). Στο πρώτο κβάντο γίνεται turn=1 στο δεύτερο turn=0, άρα η Δ1 μπαίνει στο ΚΤ σε χρόνο **200** και η Δ0 σε χρόνο **300** μονάδες.

Σεπτέμβριος 2020

Θέμα 4^{ο} (15 μονάδες)

Δίνεται ένα σύστημα Linux και 4 διεργασίες Α, Β, Γ, Δ με αριθμούς προτεραιοτήτων 100+Χ, 102+Χ, 117 και 125 αντίστοιχα (το Χ είναι από τον ΑΜ σας). Οι διεργασίες κατέφθασαν σε χρόνο t=0 με αλφαβητική σειρά. Να δώσετε τον χρόνο τερματισμού κάθε διεργασίας αν καθεμία από αυτές έχει Χρόνο Εκτέλεσης (RUN TIME) 1000ms. Ο τρόπος υπολογισμού των κβάντων ακολουθεί την πολιτική του O(1) χρονοδρομολογητή και οι διεργασίες είναι CPU bound (δεν κάνουν I/O).

Λύση

Τα κβάντα των διεργασιών είναι: Α: (800-20Χ) ms, Β: (760-20Χ) ms, Γ:460 ms, Δ:75 ms.
Δίνοντας κβάντα με τη σειρά προτεραιοτήτων, οι χρόνοι τερματισμού είναι οι παρακάτω:
Ο χρόνος της Α (ανάλογα με το Χ):

Χ	Α
0	2295
1	2285
2	2275
3	2265
4	2255
5	2245
6	2235
7	2225
8	2215
9	2205

Για τις υπόλοιπες οι χρόνοι είναι: Β:**2535ms**, Γ:**3150 ms**, και Δ:**4000 ms**

Σεπτέμβριος 2020

Θέμα 5^{ο} (20 μονάδες)

Ένα σύστημα χρησιμοποιεί έναν αλγόριθμο με προτεραιότητες (priorities) γιά την χρονοδρομολόγηση, σε συνδυασμό με τον εκ περιτροπής (RR). Δηλαδή μόνο οι διεργασίες που έχουν την υψηλότερη προτεραιότητα και είναι «έτοιμες», μοιράζονται κάθε φορά τον χρόνο της CPU. Το σύστημα στον χρόνο 0 είναι κενό (idle). Έρχονται πέντε εργασίες για εκτέλεση με τα παρακάτω χαρακτηριστικά: (το Υ είναι από τον ΑΜ σας)

Διεργασία	Priority	Χρόνος Άφιξης	Χρόνος Εκτέλεσης
1	1	0 sec	60 sec
2	3	10Y sec	50 sec
3	3	60 sec	40 sec
4	3	80 sec	30 sec
5	1	20 sec	20 sec

Δώστε το χρονοδιάγραμμα εκτέλεσης των διεργασιών καθώς και τους Average Turnaround Time (Μέσο Χρόνο Επιστροφής) και Average Weighted Turnaround Time AWTT (Μέσο Σταθμισμένο Χρόνο Επιστροφής).

Σημείωση: Για τις προτεραιότητες, μεγαλύτερο νούμερο σημαίνει μεγαλύτερη προτεραιότητα.

Λύση

Το χρονοδιάγραμμα για Υ=9 είναι το παρακάτω. Αντίστοιχα είναι και γι άλλες τιμές του Υ.

	10	20	30	40	50	60	70	80	90	100	110	120	130	135	140	150	155	160	165	170	180	190	200
1	10	10	5	5	5	5																10	10
2										3.333	3.333	3.333	3.333	1.666	2.5	5	2.5	5	5	5	10		
3							10	10	5	3.333	3.333	3.333	3.333	1.666									
4									5	3.333	3.333	3.333	3.333	1.666	2.5	5	2.5						
5			5	5	5	5																	

Επομένως οι χρόνοι είναι:

X	ATT	AWTT
0	100	2.09
1	102	3
2	112	3.31
3	120	3.5
4	128	3.7
5	136	3.9
6	112	2.7
7	106	2.55
8	100	2.4
9	96	2.30

Θέμα 6ο (15 μονάδες)

Θεωρήστε ένα σύστημα με 4 ίδιους πόρους που μοιράζονται σε 3 διεργασίες. Κάθε διεργασία απαιτεί όχι περισσότερους από 2 πόρους. Όταν μία διεργασία ζητά έναν πόρο, το λειτουργικό σύστημα δίνει οποιονδήποτε διαθέσιμο πόρο. Αν κανένας πόρος δεν είναι διαθέσιμος, η διεργασία περιμένει.

α) Είναι δυνατόν να έχουμε αδιέξοδο στο σύστημα αυτό; Αν ναι, δώστε μια σειρά αιτήσεων που καταλήγει σε αδιέξοδο. Αν όχι, ποιές από τις 4 συνθήκες αναγκαίες για αδιέξοδο δεν υπάρχουν στο σύστημα αυτό;

β) Απαντήστε πάλι στην ερώτηση (α) υποθέτοντας ότι κάθε διεργασία χρειάζεται όχι περισσότερους από τρεις πόρους.

Λύση

α) **ΟΧΙ**, γιατί αν λάβει κάθε διεργασία από έναν πόρο περισσεύει ένας. Οποιαδήποτε διεργασία τον πάρει αυτόν μπορεί να τελειώσει, άρα επιστρέφει και τους δύο που έχει. Με 4 πόρους μπορούν να τελειώσουν και οι υπόλοιπες δύο διεργασίες.

β) **ΝΑΙ**. Παράδειγμα αιτήσεων: P1,P1,P2,P2,P1,P2 και οι δύο διεργασίες είναι σε αδιέξοδο. Άλλο παράδειγμα: P1,P2,P3,P1,P2,P3,P1, όπου είναι σε αδιέξοδο και οι τρεις διεργασίες.

Ιούνιος 2020

Θέμα 1ο (30 μονάδες)

Σε ένα σύστημα που στον χρόνο 0 είναι αδρανές, έρχονται τρία jobs για εκτέλεση με τα παρακάτω χαρακτηριστικά:

Job	Χρόνος άφιξης	Χρόνος εκτέλεσης
1	0 sec	100 sec
2	30 sec	10+10X sec
3	90 sec	50 sec

Υπολογίστε τους μέσους χρόνους επιστροφής (Average Turnaround Time), μέσους ανηγμένους χρόνους επιστροφής (Average Weighted Turnaround Time), και μέσους χρόνους αναμονής (Average Waiting Time) αν το σύστημα χρησιμοποιεί τους αλγόριθμους:

α) FCFS (First Come First Served)

β) SRTN (Shortest Remaining Time Next).

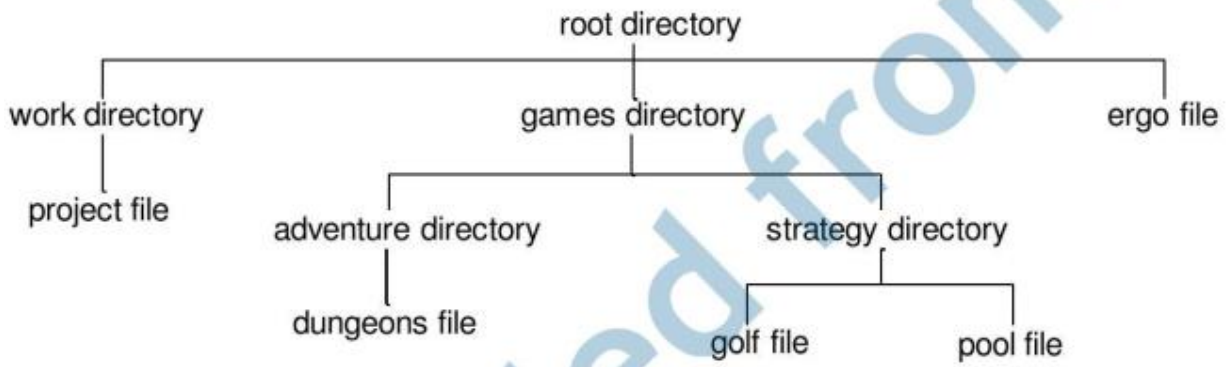
Ενδεικτική λύση

X	FCFS			SRTN		
	ATT	AWTT	AWT	ATT	AWTT	AWT
0	83.33	3.47	30.00	63.33	1.17	10.00
1	90.00	2.37	33.33	73.33	1.27	16.67
2	96.67	2.04	36.67	83.33	1.37	23.33
3	103.33	1.92	40.00	93.33	1.47	30.00
4	110.00	1.87	43.33	100.00	1.33	33.33
5	116.67	1.86	46.67	106.67	1.37	36.67
6	123.33	1.87	50.00	116.67	1.64	43.33
7	130.00	1.89	53.33	120.00	1.57	43.33
8	136.67	1.93	56.67	123.33	1.51	43.33
9	143.33	1.97	60.00	126.67	1.47	43.33
100	150.00	2.01	63.33	130.00	1.43	43.33
101	150.67	2.02	63.67	130.33	1.43	43.33
102	151.33	2.02	64.00	130.67	1.42	43.33
103	152.00	2.03	64.33	131.00	1.42	43.33
104	152.67	2.03	64.67	131.33	1.42	43.33
105	153.33	2.04	65.00	131.67	1.41	43.33
106	154.00	2.04	65.33	132.00	1.41	43.33
107	154.67	2.05	65.67	132.33	1.41	43.33
108	155.33	2.05	66.00	132.67	1.41	43.33
109	156.00	2.06	66.33	133.00	1.40	43.33

Ιούνιος 2020

Θέμα 2ο (20 μονάδες)

Να δώσετε τα i-nodes και data blocks για το παρακάτω δένδρο (καθορίστε εσείς τα νούμερα των i-nodes και data blocks):



Ενδεικτική λύση

Ορισμένα i-nodes και μπλοκς είναι τα ακόλουθα:

root dir	
1	.
1	..
x	work
y	games
z	ergo

i-node x
type:dir
…
w

block w	
x	.
1	..
a	project

i-node a
type:file
b

block b
…

i-node y
type:dir
…
c

block c	
y	.
1	..
d	adventure
e	strategy

i-node d
type:dir
…
f

block f	
d	.
c	..
g	dumgeons

i-node e
type:dir
h

…

Όπου x,y,x,w,a,b,c … θα πρέπει να βάλετε δικούς σας αριθμούς.

Downloaded from

Ιούνιος 2020

Θέμα 3ο (20 μονάδες)

Έστω ότι δίνεται μια επέκταση του προβλήματος του παραγωγού καταναλωτή με δύο παραγωγούς και δύο καταναλωτές που εκτελούνται ασύγχρονα και ταυτόχρονα. Το σύστημα έχει τους παρακάτω περιορισμούς:

1. Κάθε διεργασία έχει ένα κρίσιμο τμήμα που τοποθετεί ή εξάγει ένα στοιχείο από έναν άπειρο απομονωτή.
2. Οι καταναλωτές δεν μπορούν να επανεισέλθουν (loop) στο κρίσιμο τμήμα τους πριν γράψουν (κάθε φορά) και οι δύο παραγωγοί.
3. Κανένας από τους παραγωγούς δεν μπορεί να επανεισέλθει (loop) στο κρίσιμο τμήμα του πριν διαβάσουν και οι δύο καταναλωτές. Εξαίρεση αποτελεί το πρώτο γράψιμο των παραγωγών.
4. Και οι δύο παραγωγοί μπορούν να γράφουν ταυτόχρονα. Και οι δύο καταναλωτές μπορούν να διαβάζουν ταυτόχρονα.

Χρησιμοποιείστε semaphores για να επιτύχετε αμοιβαίο αποκλεισμό και συγχρονισμό των διεργασιών.

Ενδεικτική λύση

```
Semaphore P1=2, P2=2, C1=0, C2=0

Process Producer1     Process Producer2     Process Consumer1     Process Consumer2

loop                  loop                  loop                  loop
…                     …                     …                     …
down(P1)              down(P2)              down(C1)              down(C2)
down(P1)              down(P2)              down(C1)              down(C2)
produce               produce               consume               consume
up (C1)               up (C1)               up (P1)               up (P1)
up (C2)               up (C2)               up (P2)               up (P2)
…                     …                     …                     …
until forever         until forever         until forever         until forever
```

Θέμα 4ο (15 μονάδες)

Δίδεται η παρακάτω κατάσταση ενός συστήματος:

	Δοσμένοι πόροι				Μέγιστο αιτήσεων				Διαθέσιμοι πόροι			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	X	0				
P_2	1	3	Y	4	2	3	Y	6				
P_3	0	6	0	2	0	6	X	2				
P_4	0	0	1	4	4	Y	5	6				

Απαντήστε στις παρακάτω ερωτήσεις χρησιμοποιώντας τον αλγόριθμο αποφυγής αδιεξόδου:

α. Ποιά είναι τα περιεχόμενα του πίνακα "Απομένουσες αιτήσεις"

β. Είναι το σύστημα σε αδιέξοδο;

γ. Αν έλθει από τη διαδικασία P_1 η αίτηση (0,X,2,0), μπορεί να εξυπηρετηθεί αμέσως;

Ενδεικτική λύση

α. Οι απομένουσες αιτήσεις προκύπτουν από αφαίρεση των δοσμένων πόρων από τις μέγιστες αιτήσεις:

	Απομένουσες αιτήσεις			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	X	0
P_2	1	0	0	2
P_3	0	0	X	0
P_4	4	Y	5	6

όπου στα X και Y θα υπάρχουν οι αντίστοιχοι δικοί σας αριθμοί.

β) Παρότι χρησιμοποιείται ο αλγόριθμος αποφυγής αδιεξόδου, η ερώτηση είναι αν το σύστημα για κάποιον λόγο είναι σε αδιέξοδο. Χρησιμοποιούμε τον έλεγχο ασφαλείας για να το ελέγξουμε. Η παρατήρηση ότι η P_4 στην ουσία έχει παράνομες μέγιστες αιτήσεις (αφού δεν υπάρχουν 4 πόροι A στο σύστημα) θεωρείται σωστή απάντηση.

γ) Προσποιούμαστε ότι δίνουμε τους πόρους (εφ΄ όσον X<6) και κάνουμε έλεγχο ασφαλείας.

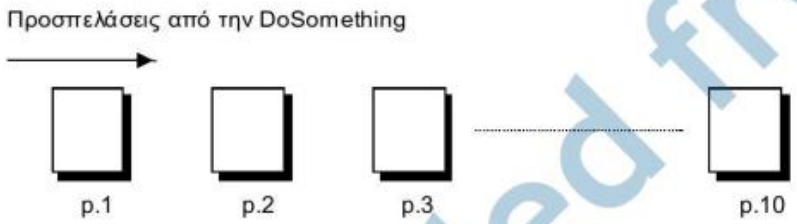
Θέμα 4ο (15 μονάδες)
Δίδεται ο παρακάτω κώδικας:

Repeat

DoSomething;

Until Done

Η διεργασία "DoSomething" κάνει διαδοχική προσπέλαση στις ίδιες πάντα 10 σελίδες ιδεατής μνήμης σε κάθε κλήση της. Υποθέστε ότι στη διεργασία δίνονται 6 πλαίσια της κυρίως μνήμης. και ότι καμία από τις δέκα σελίδες δεν είναι στη μνήμη αρχικά.



α) Ποιά είναι η σχέση ανάμεσα στον αριθμό των σφαλμάτων σελίδας και στον αριθμό των κλήσεων της διεργασίας "DoSomething" όταν χρησιμοποιείται ο αλγόριθμος FIFO ή όταν χρησιμοποιείται ο αλγόριθμος LRU;

Ενδεικτική απάντηση

Επειδή όλες οι σελίδες της διεργασίας δεν χωρούν στην μνήμη, και επιπλέον γίνεται γραμμική προσπέλασή τους, σε κάθε περίπτωση ο αριθμός των σφαλμάτων είναι:

$\Sigma\Sigma=10n$

όπου n είναι ο αριθμός κλήσεων της διεργασίας.

Θέμα 1ο (15 μονάδες)

Δίδεται ένα σύστημα που υποστηρίζει 5000 χρήστες. Υποθέστε ότι θέλετε να επιτρέψετε σε 4990 από αυτούς τους χρήστες να έχουν προσπέλαση σε κάποιο αρχείο.

α) Πως θα προσδιορίσετε αυτό το σχήμα προστασίας στο UNIX;

β) Μπορείτε να προτείνετε ένα άλλο σχήμα προστασίας που να είναι πιο αποτελεσματικό στην περίπτωση αυτή, απ' ότι το σχήμα που προσφέρει το UNIX;

Λύση:

α) Κάνουμε ένα group και προσθέτουμε σε αυτό έναν έναν και τους 4990 χρήστες.Έπειτα δίνουμε στο group το δικαίωμα πρόσβασης στο αρχείο.

β) ACL.Δίνουμε σε όλους τους χρήστες πρόσβαση το αρχείο και δημιουρώ ένα group.Στο group προσθέτω τα 10 άτομα και αποκλείω το δικαίωμα της πρόσβασης στο αρχείο.

Θέμα 2ο (15 μονάδες)

Το πρόβλημα αυτό βασίζεται στο παράδειγμα που σας δόθηκε σε φωτοτυπίες, για concurrent I/O με δύο interrupt service routines για πληκτρολόγιο και οθόνη. Στο παράδειγμα υπάρχουν τρείς εκδόσεις του κυρίως προγράμματος, η τρίτη από τις οποίες είναι η εξής:

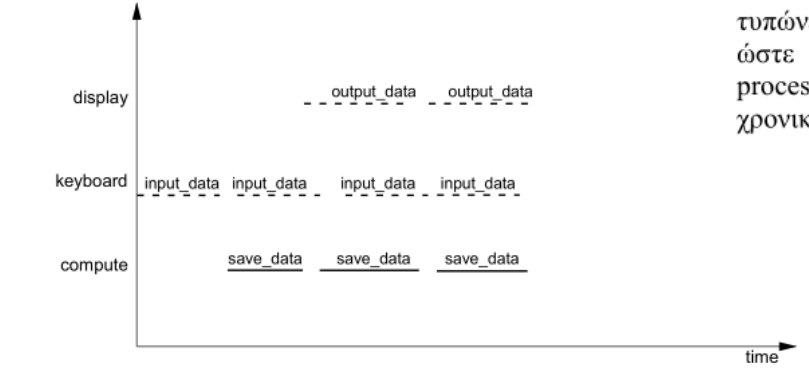
```
{MAIN - version 3}
begin {concurrent I/O}
  ioport_command:= kbd_and_dsp_interrupts_disable;
  initialize_hardware;
  connect_interrupt_vectors(kbd,dsp);
  outdone:=true;
  start_input;
  while true do
    begin
      while not indone do {keeptesting};
      copy (indata, save_data);
      save_count:=incount;
      start_input;
      while not outdone do {keeptesting};
      copy(save_data,outdata);
      start_output;
      process_data;
    end {while};
end {concurrent I/O};
```

Όταν τελειώσει μια γραμμή από το πληκτρολόγιο, το interrupt service routine του πληκτρολογίου κάνει την μεταβλητή indone=true. Αντίστοιχα, όταν τυπωθούν τα δεδομένα το interrupt service routine της οθόνης κάνει την μεταβλητή outdone=true. Παρατηρούμε ότι το πρόγραμμα αυτό τυπώνει την γραμμή του γράφτηκε στο πληκτρολόγιο. Ξαναγράψτε το κυρίως πρόγραμμα έτσι ώστε αυτά που τυπώνονται στην οθόνη να είναι τα αποτελέσματα του process_data. Το process_data φυσικά χρησιμοποιεί ως είσοδο αυτό που δόθηκε απο το πληκτρολόγιο. Δώστε το χρονικό διάγραμμα σημειώνοντας επάνω σ' αυτό ποιό buffer χρησιμοποιεί το κάθε διεργασία.

Λύση:

Θέμα 2 Μονάδες 15

```
{MAIN - version 4}
begin {concurrent I/O}
   ioport_command:= kbd_and_dsp_interrupts_disable;
   initialize_hardware;
   connect_interrupt_vectors(kbd,dsp);
   outdone:=true;
   start_input;
   while true do
   begin
      while not indone do {keeptesting};
      copy (indata, save_data);
      save_count:=incount;
      start_input;
      process_data;
      while not outdone do {keeptesting};
      copy(save_data,outdata);
      start_output;
   end {while};
end {concurrent I/O};
```



Φεβρουάριος 2021

Θέμα 3ο (20 μονάδες)

Οι ακόλουθες διεργασίες έρχονται ταυτόχρονα για εκτέλεση και αμέσως συναγωνίζονται για την CPU. (το Υ είναι από τον ΑΜ σας)

Αριθμός διεργασίας	Χρόνος εκτέλεσης
1	50
2	40
3	20
4	10+10Υ

Να εξετάσετε τους εξής αλγόριθμους χρονοδρομολόγησης:

α) SRTN (Shortest Remaining Time Next). Στην περίπτωση ίσων απομενόντων χρόνων η χρονοδρομολόγηση γίνεται με αριθμητική σειρά

β) FIFO (First-In, First-Out). Στην περίπτωση ταυτόχρονων αφίξεων η χρονοδρομολόγηση γίνεται με αριθμητική σειρά

γ) Round Robin με κβάντο = 10 (αρχίστε τη δρομολόγηση με αριθμητική σειρά)

δ) Round Robin με κβάντο = 20 (αρχίστε τη δρομολόγηση με αριθμητική σειρά)

Για κάθε περίπτωση υπολογίστε:

α) Τον χρόνο επιστροφής κάθε διεργασίας

β) Τον μέσο χρόνο επιστροφής

Λύση:

Θέμα 3 Μονάδες 20 (Υ= τελευταίο ψηφίο)

Μέσοι χρόνοι

Υ	SRTN	FCFS	RR10	RR20
0	57.5	92.5	85	90
1	67.5	95	100	97.5
2	75	97.5	112.5	112.5
3	82.5	100	122.5	117.5
4	87.5	102.5	127.5	122.5
5	90	105	130	125
6	92.5	107.5	132.5	127.5
7	95	110	135	130
8	97.5	112.5	137.5	132.5
9	100	115	140	135

Φεβρουάριος 2021

Θέμα 4ο (20 μονάδες)

Μελετήστε την παρακάτω λύση στο πρόβλημα του αμοιβαίου αποκλεισμού 2 διεργασιών:

Διεργασία 1	Διεργασία 2
. . .	. . .

Begin {BeginExcusion}
Flag[i] = True;
While Turn <> i **Do**
Begin
If flag[j] **Then** Wait;
Turn = i;
End {While}
End; {BeginExcusion}
critical section
Begin {EndExclusion}
Flag[i] = False;
Resume the other process;
End; {EndExclusion}
	Begin {BeginExcusion}
	Flag[i] = True;
	While Turn <> i **Do**
	Begin
	If flag[j] **Then** Wait;
	Turn = i;
	End {While}
	End; {BeginExcusion}
	critical section
	Begin {EndExclusion}
	Flag[i] = False;
	Resume the other process;
	End; {EndExclusion}

Υποθέστε οτι τα παρακάτω είναι κοινά (global) και για τις δύο διεργασίες:

Flag : array[1. .2] of Boolean; {Αρχικά False}
Turn : 1. .2 {Αρχικά 1 ή 2}

Υποθέστε οτι για κάθε διεργασία, τα i και j είναι τοπικά (local variables). Στην διεργασία 1, i=1 και j=2. Στην διεργασία 2, i=2 και j=1. Τι μπορεί να πάει στραβά; Εξηγήστε επακριβώς το πρόβλημα.

Λύση:

Θέμα 4 Μονάδες 20

Διεργασία 1	Διεργασία 2
. . .	. . .

Begin {BeginExcusion}
Flag[i] = True;
While Turn <> i **Do**
Begin
If flag[j] **Then** Wait;
Turn = i;
End {While}
End; {BeginExcusion}
critical section
Begin {EndExclusion}
Flag[i] = False;
Resume the other process;
End; {EndExclusion}
	Begin {BeginExcusion}
	Flag[i] = True;
	While Turn <> i **Do**
	Begin
	If flag[j] **Then** Wait;
	Turn = i;
	End {While}
	End; {BeginExcusion}
	critical section
	Begin {EndExclusion}
	Flag[i] = False;
	Resume the other process;
	End; {EndExclusion}

Αρχίζει ο turn<>i. Κόβεται αμέσως πριν το Turn=i. Μπαίνει στο κρίσιμο τμήμα ο άλλος και κόβεται Μέσα στο κρίσιμο τμήμα. Μπαίνει και ο πρώτος. Και οι δύο στο κρίσιμο τμήμα.

Θέμα 5ο (15 μονάδες)

Ένα σύστημα δυναμικής κατανομής της μνήμης περιέχει τα παρακάτω μεγέθη κενών (ελεύθερων) τμημάτων κατά σειρά: 8KB, 5KB, 18KB, 16KB, 6KB, 7KB, 10KB, και 13KB. Ποιά ελεύθερα τμήματα θα χρησιμοποιηθούν αν γίνονται οι παρακάτω διαδοχικές αιτήσεις για μεγέθη τμημάτων (το X είναι από τον ΑΜ σας):

1) 12 KB
2) 1+X KB
3) 9 KB

και χρησιμοποιούνται οι αλγόριθμοι:

α) first fit
β) best fit
γ) worst fit
δ) next fit

Λύση:

Θέμα 5 Μονάδες 15

X<5 12 1-5 9

	FF	BF	WF	NF
8	1-5			
5		1-5		
18	12		12	12
16	9		1-5	1-5
6				
7				
10		9		9
13		12	9	

X=5 12 6 9

	FF	BF	WF	NF
8	6			
5				
18	12		12	12
16	9		6	6
6		6		
7				
10		9		9
13		12	9	

X=6 12 7 9

	FF	BF	WF	NF
8	7			
5				
18	12		12	12
16	9		7	7
6				
7		7		
10		9		9
13		12	9	

X=7 12 8 9

	FF	BF	WF	NF
8	8	8		
5				
18	12		12	12
16	9		8	8
6				
7				
10		9		9
13		12	9	

X=8 12 9 9

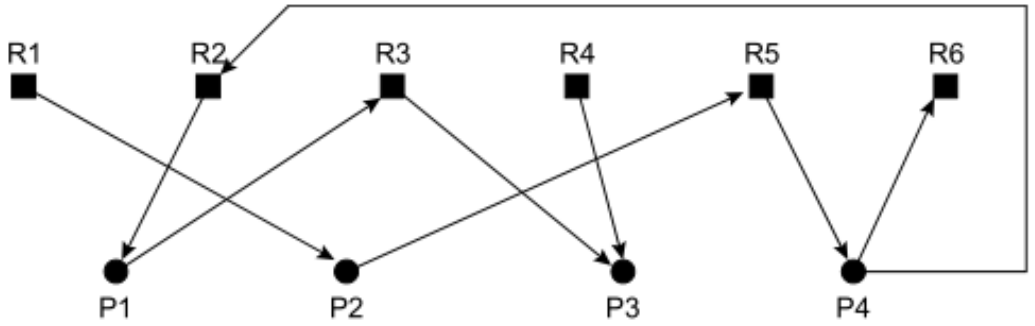
	FF	BF	WF	NF
8				
5				
18	12		12	12
16	9	9	9	9
6				
7				
10	9	9		9
13		12	9	

X=9 12 10 9

	FF	BF	WF	NF
8				
5				
18	12		12	12
16	10	9	10	10
6				
7				
10	9	10		9
13		12	9	

Θέμα 6ο (15 μονάδες)

Δίδεται το παρακάτω γράφημα κατανομής πόρων. Δώστε ένα παράδειγμα αίτησης που θα προκαλέσει αδιέξοδο και στις τέσσερις διεργασίες. Δώστε επίσης ένα παράδειγμα αίτησης που θα προκαλέσει αδιέξοδο σε δύο μόνο διεργασίες.



Λύση:

Θέμα 6 Μονάδες 15

α) P3→R1

β) P4→R1, ή P3→R2, ή P1→R5

Σεπτέμβριος 2021

Θέμα 1ο (20 μονάδες)

Θεωρήστε την παρακάτω λογισμική λύση στο πρόβλημα του αμοιβαίου αποκλεισμού για δύο διεργασίες:

```
Type
      Turntype=(One, Two, Either);
Var
      Turn : Turntype; {Κοινή μεταβλητή και για τις δύο διεργασίες}
Processi { το i είναι 1 ή 2}
      :
Var
Ours, Theirs : Turntype; {Τοπική στην processi. Στην διεργασία 1, Ours=One και Theirs=Two.
      Στην διεργασία 2 Ours=Two και Theirs=One}
Begin {Processi}
      :
      Begin {Αρχή Αποκλεισμού}
      If Turn=Theirs Then
            Wait
      Else
            Turn=Ours;
      End; {Αρχής Αποκλεισμού}
:
Κρίσιμη περιοχή
:
      Begin {Τέλος Αποκλεισμού}
      If other process is suspended Then
            Begin
                  Turn=Theirs
                  resume other process
            End
      Else
            Turn=Either;
      End {Τέλους Αποκλεισμού}
      :
End. {processi}
```

Η ιδέα είναι να επιτρέψουμε είσοδο στο κρίσιμο τμήμα μόνο όταν είναι η "σειρά μας" (Our turn) ή "οποιουδήποτε σειρά" (either turn). Η τιμή "Either" καταργεί την απαίτηση να εναλλάσσονται οι διεργασίες που θα μπαίνουν στο κρίσιμο τμήμα τους. Αρχικά, Turn=Either. Δουλεύει; Αν ΝΑΙ γιατί; Αν ΟΧΙ γιατί;

Λύση:

Σεπτέμβριος 2021

Θέμα 2ο (20 μονάδες)

Σε ένα σύστημα έρχονται τέσσερα jobs για εκτέλεση με τα παρακάτω χαρακτηριστικά:

Job	Χρόνος άφιξης	Χρόνος εκτέλεσης
1	10X sec	40 sec
2	40 sec	50 sec
3	90 sec	60 sec
4	140 sec	70 sec

όπου X είναι το προτελευταίο ψηφίο του αριθμού μητρώου σας.

Υπολογίστε το μέσο χρόνο επιστροφής (Average Turnaround Time), μέσο ανηγμένο χρόνο επιστροφής (Average Weighted Turnaround Time), μέσο χρόνο αναμονής (Average Waiting Time) και μέσο ανηγμένο χρόνο αναμονής (Average Weighted Waiting) αν το σύστημα χρησιμοποιεί τους αλγόριθμους FCFS (First Come First Served) και Συντομότερος Εναπομένων Χρόνος Πρώτα (SRTN: Shortest Remaining Time Next). Αγνοήστε τον χρόνο μεταγωγής διεργασιών (Task Switching Time).

Σημείωση: Σε ταυτόχρονη άφιξη ξεκινά η διεργασία με μικρότερο αύξοντα αριθμό. Σε περίπτωση ίσων απομενόντων χρόνων εκτελείται η διεργασία με τον μικρότερο αύξοντα αριθμό.

Λύση:

Σεπτέμβριος 2021

Θέμα 3ο (15 μονάδες)

Στον οδηγό δίσκου φθάνουν αιτήσεις για τους κυλίνδρους 7, 20, 10, 36, Υ, 22, και 42, με αυτή τη σειρά (όπου Υ είναι το τελευταίο ψηφίο του αριθμού μητρώου σας). Η μετακίνηση της κεφαλής σε γειτονικό κύλινδρο χρειάζεται 4 msec. Πόσος συνολικός χρόνος αναζήτησης απαιτείται με τους παρακάτω αλγορίθμους;

α) Πρώτη εισερχόμενη - Πρώτη Εξυπηρετούμενη.

β) Εγγύτερος κύλινδρος πρώτος.

γ) Αλγόριθμος του ανελκυστήρα (αρχική κατεύθυνση ΑΝΩ).

Σε όλες τις περιπτώσεις, ο βραχίονας βρίσκεται αρχικά στον κύλινδρο 20.

Λύση:

Σεπτέμβριος 2021

Θέμα 4^{ο} (15 μονάδες)

Ένα λειτουργικό σύστημα χρησιμοποιεί ιδεατή μνήμη κατ' απαίτηση (virtual memory με demand paging). Δηλαδή, μια σελίδα δεν έρχεται στην μνήμη αν δεν ζητηθεί. Έτσι, αρχικά δεν φορτώνεται καμιά σελίδα στην μνήμη και έχουμε σφάλμα σελίδας ακόμα και με την πρώτη διεύθυνση που ζητείται. Το σύστημα χρησιμοποιεί τον αλγόριθμο FIFO για την αντικατάσταση των σελίδων. Δίνεται η παρακάτω σειρά αιτήσεων σελίδων (αριθμοί σελίδων που ζητούνται, όπου X είναι το προτελευταίο ψηφίο του αριθμού μητρώου σας):

1,2,3,4,X,3,X,3,0,2,4,1,X,4,5,X,2,3

Πόσα page faults έχουμε με την σειρά αυτή (μετρώντας και τα αρχικά) αν:

α) Δίνονται στην διεργασία τρία πλαίσια κύριας μνήμης

β) Δίνονται στην διεργασία τέσσερα πλαίσια κύριας μνήμης

γ) Δίνονται στην διεργασία τρία πλαίσια και χρησιμοποιείται ο αλγόριθμος LRU.

Λύση:

Σεπτέμβριος 2021

Θέμα 5ο (15 μονάδες)

Δίνεται ένα σύστημα Linux και 4 διεργασίες Α, Β, Γ, Δ με αριθμούς προτεραιοτήτων 101, 102+Υ, 112+Υ και 122 αντίστοιχα (το Υ είναι το τελευταίο ψηφίο του αριθμού μητρώου σας). Οι διεργασίες κατέφθασαν σε χρόνο t=0 με αλφαβητική σειρά. Να δώσετε τον **χρόνο τερματισμού** κάθε διεργασίας αν καθεμία από αυτές έχει Χρόνο Εκτέλεσης (RUN TIME) 1000ms. Ο τρόπος υπολογισμού των κβάντων ακολουθεί την πολιτική του Ο(1) χρονοδρομολογητή και οι διεργασίες είναι CPU bound (δεν κάνουν I/O).
Σημείωση: Υ είναι το τελευταίο ψηφίο του αριθμού μητρώου σας.

Λύση:

Σεπτέμβριος 2021

Θέμα 6ο (15 μονάδες)

Δίδεται η παρακάτω κατάσταση ενός συστήματος:

	Δοσμένοι πόροι				Μέγιστο αιτήσεων				Διαθέσιμοι πόροι			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	1	0	1	1	2	4	Y	0	1
P_1	1	0	1	2	1	Y	2	3				
P_2	1	Y	3	0	2	Y	3	7				
P_3	0	2	7	2	0	3	9	2				
P_4	3	0	1	4	9	3	1	6				

Απαντήστε στις παρακάτω ερωτήσεις χρησιμοποιώντας τον αλγόριθμο αποφυγής αδιεξόδου:

α. Ποιά είναι τα περιεχόμενα του πίνακα "Απομένουσες αιτήσεις"

β. Είναι το σύστημα σε αδιέξοδο;

γ. Αν έλθει από τη διαδικασία P_1 η αίτηση (0,Y,0,1), μπορεί να εξυπηρετηθεί σύμφωνα με τον αλγόριθμο;

Σημείωση: Υ είναι το τελευταίο ψηφίο του αριθμού μητρώου σας.

Λύση:

Σεπτέμβριος 2022

Θέμα 1ο (15 μονάδες)

Ένα υπολογιστικό σύστημα διαθέτει 4MBytes μνήμη για τις διεργασίες. Το λειτουργικό σύστημα διαχειρίζεται τη μνήμη με πλαίσια σταθερού μεγέθους. Ένας πίνακας πλαισίων, που αποθηκεύεται σ` αυτά τα 4MBytes μνήμης, περιέχει την κατάσταση κάθε πλαισίου της μνήμης. Θέλουμε να υπολογίσουμε πόσο μεγάλο θα πρέπει να είναι κάθε πλαίσιο;

Πίνακας Πλαισίων	Διαθέσιμος χώρος για διεργασίες

4MByte Μνήμη

Οι σχεδιαστές αποφάσισαν ότι το μέγεθος των πλαισίων μπορεί να είναι είτε 2K, είτε 4K, είτε 8K bytes. Ποια από αυτές τις επιλογές θα ελαχιστοποιήσει τον χώρο που δεν χρησιμοποιείται από τις διεργασίες επειδή είτε καταλαμβάνεται από τον πίνακα πλαισίων είτε χάνεται λόγω κατάτμησης (fragmentation); Υποθέστε ότι :

- Κατά μέσο όρο 10 διεργασίες βρίσκονται στην μνήμη. Ο μέσος χώρος που χάνεται λόγω κατάτμησης είναι ½ πλαίσιο για κάθε διεργασία.
- Ο πίνακας πλαισίων έχει μια εγγραφή για κάθε πλαίσιο και κάθε εγγραφή χρειάζεται 10 bytes.

Λύση:

Σεπτέμβριος 2022

Θέμα 2ο (24 μονάδες)

Δίνεται σύστημα Linux και 4 διεργασίες Α,Β,Γ,Δ με αριθμούς προτεραιοτήτων 103, 107, 116 και 127 αντίστοιχα. Οι διεργασίες κατέφθασαν σε χρόνο t=0 με αλφαβητική σειρά. Να δώσετε τον **χρόνο τερματισμού** κάθε διεργασίας αν καθεμία από αυτές έχει Χρόνο Εκτέλεσης (RUN TIME) 1500ms .Ο τρόπος υπολογισμού των κβάντων ακολουθεί την πολιτική του Ο(1) χρονοδρομολογητή και οι διεργασίες είναι CPU bound (δεν κάνουν I/O).

Λύση:

Σεπτέμβριος 2022

Θέμα 3ο (15 μονάδες)

Η παρακολούθηση του ελεύθερου χώρου στο δίσκο μπορεί να γίνει με τη χρήση μιας λίστας ελεύθερων μπλοκ ή ενός χάρτη ψηφίων. Οι διευθύνσεις του δίσκου απαιτούν *Δ* bits. Αν ένας δίσκος έχει χωρητικότητα *M* μπλοκ από τα οποία Ε είναι ελεύθερα, δώστε τη συνθήκη με την οποία η λίστα ελευθέρων μπλοκ χρησιμοποιεί λιγότερο χώρο από το χάρτη ψηφίων. Αν το *Δ* έχει τιμή 16 bits, εκφράστε την απάντησή σας ως ποσοστό επί τοις εκατό του χώρου δίσκου ο οποίος πρέπει να είναι ελεύθερος για να ισχύει η συνθήκη.

Λύση:

Σεπτέμβριος 2022

Θέμα 4ο (26 μονάδες)

A. Σε υπολογιστές που διαθέτουν την εντολή TSL (Test and Set Lock), είδαμε ότι μπορεί να γραφεί μια ρουτίνα εισόδου σε κρίσιμο τμήμα ως εξής :

```
enter :
    TSL REGISTER, LOCK
    CMP REGISTER, #0
    JNE, enter
    RET
```

Έστω τώρα ότι ο υπολογιστής μας δεν διαθέτει την εντολή TSL, αλλά διαθέτει μια εντολή που ανταλλάσσει τα περιεχόμενα ενός καταχωρητή με τα περιεχόμενα μιας θέσης μνήμης σε αδιαίρετη ενέργεια. Μπορεί αυτή η εντολή να χρησιμοποιηθεί για την ρουτίνα enter; Αν ναι γράψτε την ρουτίνα αυτή.

B. Ένα random access αρχείο έχει records καθορισμένου μεγέθους αλλά χρησιμοποιείται σειριακά από διάφορες διεργασίες που γράφουν στο αρχείο αυτό. Κάθε διεργασία γράφει στο επόμενο διαθέσιμο record του αρχείου. Γράψτε τον ψευδοκώδικα των διεργασιών αυτών, χρησιμοποιώντας binary semaphores. Προσοχή: το λειτουργικό σύστημα δεν αυξάνει από μόνο του τον δείκτη του αρχείου (file pointer). Δεν υπάρχουν processes που να διαβάζουν από το αρχείο

Λύση:

Σεπτέμβριος 2022

Θέμα 5ο (20 μονάδες)

Δίδεται ένα σύστημα που αποτελείται από 5 πόρους του ιδίου τύπου οι οποίοι διαμοιράζονται σε 4 διεργασίες. Οι πόροι ζητούνται και αποδίδονται από τις διεργασίες ένας κάθε φορά (δηλαδή μόνο αφού δοθεί ένας πόρος μπορεί η διεργασία να ζητήσει τον επόμενο). Να αποδείξετε ότι το σύστημα δεν μπορεί να φθάσει σε αδιέξοδο αν ισχύουν οι εξής δυο συνθήκες:

1. Ο μέγιστος αριθμός πόρων που μπορεί να ζητήσει κάθε διεργασία (μέγιστες ανάγκες) είναι ανάμεσα στο 1 και το 5.
2. Το άθροισμα όλων των μέγιστων αναγκών είναι μικρότερο από το 9.

Λύση:

Φεβρουάριος 2023

Θέμα 1ο (20 μονάδες)

Μετρήσεις σε συγκεκριμένο σύστημα έχουν δείξει, ότι μια μέση διεργασία εκτελείται για χρόνο Τ πριν ανασταλει για είσοδο/έξοδο.Η εναλλαγή λογίζεται ως επιβάρυνση.Για χρονοπρογραμματισμό εξυπηρέτησης εκ περιτροπής με κβάντο χρόνου Κ,δώστε τον τύπο αποδοτικότητας της CPU,για καθε μια απο τις ακόλουθες περιπτώσεις:

a) $K = \infty$
b) $K>T$
c) $E<K<T$
d) $K=E<T$
e) $K \approx 0$

Λύση:

α) Αφού το κβάντο είναι τεράστιο κάθε διεργασία θα διακοπεί όταν ζητήσει I/O, δηλαδή μετά από χρόνο Τ. Επομένως: U=T/(T+E)

β) Αφού το κβάντο είναι μεγαλύτερο του χρόνου που θα τρέξει χωρίς διακοπή μια διεργασία, κάθε διεργασία θα διακοπεί όταν ζητήσει I/O, δηλαδή μετά από χρόνο Τ. Επομένως και πάλι: U=T/(T+E)

γ) Αφού το κβάντο είναι μικρότερο από Τ, κάθε διεργασία θα συμπληρώσει ένα κβάντο πριν διακοπεί. Επομένως: U=K/(K+E)

δ) Το κβάντο υποτίθεται ότι είναι μικρότερο του Τ, αλλιώς θα είχαμε την περίπτωση (β). Έχουμε λοιπόν την περίπτωση (γ) με ίσα τα Κ και Ε. Επομένως: U=50%

ε) Αφού το κβάντο είναι σχεδόν ίσο με το 0, έχουμε την περίπτωση (γ) με πάρα πολύ μικρό το Κ. Επομένως: U≈ 0

Φεβρουάριος 2023

Θέμα 2ο (25 μονάδες)

Μελετήστε το παρακάτω πρόγραμμα αμοιβαίου αποκλεισμού. Καλύπτει τις απαιτήσεις μιας "καλής" λύσης; Αν ναι, δώστε επιχειρήματα για την ορθότητά του. Αν όχι, εξηγήστε γιατί.

```
program/module mutex4
...
var p1using, p2using: boolean;

process p1;                          process p2;
  begin                                begin
    while true do                        while true do
      begin                                begin
        p1using:=true;                       p2using:=true;
        while p2using do                     while p1using do
          begin                                begin
            p1using:=false;                      p2using:=false;
            p1using:=true                        p2using:=true
          end; {while}                         end; {while}
        critical_section;                    critical_section;
        p1using:=false;                      p2using:=false;
        other_p1_processing                  other_p2_processing
  end; {p1}                            end; {p2}

{parent process}
begin {mutex4}
  p1using:=false;
  p2using:=false;
  initiate p1, p2
end {mutex4}
```

* *Initiate σημαίνει ξεκίνα τις ακόλουθες διεργασίες*

ΠΡΟΤΕΙΝΟΜΕΝΗ ΛΥΣΗ

Ο αμοιβαίος αποκλεισμός είναι μια ιδιότητα του ελέγχου συγχρονισμού, η οποία διασφαλίζει ότι πολλές διεργασίες ή νήματα δεν εισέρχονται ταυτόχρονα στα κρίσιμα τμήματα τους.Σε αυτό το συγκεκριμένο παράδειγμα, οι δύο διεργασίες p1 και p2 μοιράζονται δύο μεταβλητές **boolean p1using** και **p2using**. Αυτές οι μεταβλητές χρησιμοποιούνται για να υποδείξουν εάν μια διεργασία χρησιμοποιεί αυτήν τη στιγμή το κρίσιμο τμήμα της ή όχι. Ο αλγόριθμος λειτουργεί βάζοντας κάθε διεργασία να ορίζει τη δική της μεταβλητή **using** μεταβλητή σε **true** πριν εισέλθει στο κρίσιμο τμήμα της και, στη συνέχεια, ελέγχοντας επανειλημμένα τη μεταβλητή **using** η άλλη διεργασία μέχρι να είναι **false**. Μόλις η μεταβλητή που χρησιμοποιεί η άλλη διεργασία είναι ψευδής, η τρέχουσα διεργασία μπορεί να εισέλθει με ασφάλεια στο κρίσιμο τμήμα της. Μετά την έξοδο από το κρίσιμο τμήμα, η τρέχουσα διεργασία ορίζει τη δική της **using** μεταβλητή **back** σε **false**, επιτρέποντας στην άλλη διεργασία να εισέλθει στο κρίσιμο τμήμα της.
Ωστόσο, αυτός ο αλγόριθμος δεν εξασφαλίζει αμοιβαίο αποκλεισμό, επειδή είναι δυνατό και για τις δύο διεργασίες να ορίσουν τις μεταβλητές χρήσης τους σε **true** ταυτόχρονα, και στη συνέχεια να εισέλθουν και οι δύο στα κρίσιμα τμήματα τους ταυτόχρονα. Αυτό μπορεί να συμβεί εάν και οι δύο διεργασίες εκτελέσουν τη γραμμή κώδικα που ορίζει τη μεταβλητή **using** σε true, προτού κάποια από αυτές ελέγξει τη μεταβλητή **using** της άλλης διεργασίας. Αυτό είναι γνωστό ως κατάσταση αγώνα.

Λύσεις (2)

3) Αυστηρή εναλλαγή

```
while (true)  {
    while (turn<>0) wait
    critical_section
    turn=1
    non_critical_section
    }
while (TRUE)  {
    while (turn<>1) wait
    critical_section
    turn=0
    non_critical_section
    }
```

Φεβρουάριος 2023

Θέμα 3ο (15 μονάδες)
Σε ένα λειτουργικό συστημα Unix που χρησιμοποιεί κόμβους-δ για την κατανομή αρχειων, κάθε μπλοκ του δίσκου περιλαμβάνει είτε 4KB δεδομένων είτε 2048 δείκτες.Ποιό είναι το μέγιστο μέγεθος αρχείου στο σύστημα αυτό;

Λύση:

Φεβρουάριος 2023

Θέμα 4ο (20 μονάδες)
Σε ένα συστημα που χρησιμοποιεί ανίχνευση αδιεξόδου και επιδιόρθωση (ανάνηψη) υπάρχουν ένα (1)printer και δυο (2) drives.Σε κάποια χρονική στιγμή της λειτουργίας του μπαίνουν 3 διεργασίες για εκτέλεση.Οι τρείς αυτές διεργασίες ζητούν περιφερεικαά με την παρακάτω σειρά:

1. Process 1 printer
2. Process 2 disk
3. Process 3 disk
4. Process 2 disk
5. Process 3 printer
6. Process 1 disk

1. Εξετάστε κάθε αίτηση και αναφέρετε αν σε κάποιο σημείο νομίζετε οτι υπάρχει αδιέξοδο.
2. Αν το σύστημα χρησιμοποιεί spooling για τον printer ,ποια αμέσως επόμενη αίτηση (δλδ No7) θα προκαλέσει αδιέξοδο;

Λύση: Ας αναλύσουμε τη σειρά των αιτημάτων που γίνονται από τις διεργασίες για να προσδιορίσουμε αν υπάρχει πιθανότητα αδιεξόδου. Θα παρακολουθούμε τους πόρους που ζητούνται και αποδεσμεύονται από κάθε διαδικασία.Αρχικά, έχουμε 1 εκτυπωτή και 2 μονάδες δίσκου.1. Η διαδικασία 1 ζητά έναν εκτυπωτή. Πόροι: Εκτυπωτής (1/1) Η διαδικασία 1 αποκτά τον εκτυπωτή με επιτυχία.2. Η διαδικασία 2 ζητά μια μονάδα δίσκου. Πόροι: Εκτυπωτής (1/1), Δίσκος (1/2) Η διαδικασία 2 αποκτά μια μονάδα δίσκου με επιτυχία.3. Η διαδικασία 3 ζητά μια μονάδα δίσκου. Πόροι: Εκτυπωτής (1/1), Δίσκος (2/2) Η διαδικασία 3 αποκτά μια μονάδα δίσκου με επιτυχία.4. Η διαδικασία 2 ζητά άλλη μονάδα δίσκου. Πόροι: Εκτυπωτής (1/1), Δίσκος (2/2) Η διαδικασία 2 δεν μπορεί να αποκτήσει δεύτερη μονάδα δίσκου, καθώς υπάρχει μόνο μία διαθέσιμη. Εισέρχεται σε κατάσταση αναμονής.
Σε αυτό το σημείο, έχουμε μια πιθανή κατάσταση αδιεξόδου. Η διαδικασία 2 περιμένει για έναν πόρο (μονάδα δίσκου) που διατηρείται από τη Διεργασία 3 και η Διεργασία 3 περιμένει έναν πόρο (εκτυπωτή) που διατηρείται από τη Διεργασία 1.5. Η διαδικασία 3 ζητά έναν εκτυπωτή. Πόροι: Εκτυπωτής (0/1), Δίσκος (2/2) Η διαδικασία 3 δεν μπορεί να αποκτήσει τον εκτυπωτή, καθώς κρατιέται από τη Διεργασία 1. Εισέρχεται σε κατάσταση αναμονής.Τώρα έχουμε αδιέξοδο. Η διαδικασία 1 κρατά τον εκτυπωτή και περιμένει για μια μονάδα δίσκου (η οποία διατηρείται από τη Διεργασία 2), η Διεργασία 2 αναμένει μια μονάδα δίσκου (κρατείται από τη Διαδικασία 3) και η Διεργασία 3 περιμένει τον εκτυπωτή (διατηρείται από τη Διεργασία 1) . Αυτή η κυκλική εξάρτηση οδηγεί σε αδιέξοδο και καμία από τις διαδικασίες δεν μπορεί να προχωρήσει.
Σε αυτό το σενάριο, θα πρέπει να χρησιμοποιηθούν μηχανισμοί ανίχνευσης αδιεξόδου και ανάκτησης για την επίλυση του αδιεξόδου. Οι αλγόριθμοι ανίχνευσης αδιεξόδου, όπως το γράφημα κατανομής πόρων ή ο αλγόριθμος του Banker, μπορούν να χρησιμοποιηθούν για τον προσδιορισμό της κατάστασης αδιεξόδου. Μόλις εντοπιστούν, μπορούν να εφαρμοστούν μέτρα ανάκτησης, όπως η προκατάληψη πόρων ή ο τερματισμός της διαδικασίας, για να ξεπεραστεί το αδιέξοδο και να επιτραπεί στις διεργασίες να συνεχίσουν να εκτελούνται.

Φεβρουάριος 2023

Θέμα 5ο (20 μονάδες)

Ενα λειτουργικό συστημα χρησιμοποιεί virtual memory με demand paging.Δηλαδη,μια σελιδα δεν ερχεται στην μνημη αν δεν ζητηθει.Ετσι,αρχικα δεν φορτωενται καμια σελιδα στην μνημη και εχουμε page fault ακομα και με την πρωτη διαυθυνση που ζητιεται.Το συστημα χρησιμοποιεί το παρακατω page trace (σειρα αιτησεων σελιδων). 1,2,3,4,1,5,1,3,2,4,3,4,5,1

Δωστε τον αριθμο των σφαλματων σελιδας (μετρωντας και τα αρχικά) καθως και τα τείκά περιεχόμενα της μνημης σε καθε μια απο τις παρακατω περιπτώσεις αν:

a. Δινονται στην διεργασια τρια πλαίσια κύριας μνήμης
b. Δίνονται στην διεργασία τέσσερα πλασια κυριας μνημης:
   Τι παρατηρείτε στα α και β;
c. Δίνονται τρία πλαίσια και χρησιμοποιειται ο αλγόριθμος LRU.

Λύση:

a

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ref		1	2	3	4	1	5	1	3	2	4	3	4	5	1
f		1	2	3	4	1	5	5	3	2	4	4	4	5	1
f			1	2	3	4	1	1	5	3	2	2	2	4	5
f				1	2	3	4	4	1	5	3	3	3	2	4
hit		X	X	X	X	X	X	✓	X	X	X	✓	✓	X	X
v					1	2	3		4	1	5			3	2

- Total references: 14
- Total distinct references: 5
- Hits: 3
- Faults: 11
- **Hit rate:** 3/14 = **21.428571428571**%
- **Fault rate:** 11/14 = **78.571428571429**%

Τελευταία περιεχόμενα μνήμης 1 , 5, 4

b

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ref		1	2	3	4	1	5	1	3	2	4	3	4	5	1
f		1	2	3	4	4	5	1	1	2	2	3	4	5	1
f			1	2	3	3	4	5	5	1	1	2	3	4	5
f				1	2	2	3	4	4	5	5	1	2	3	4
f					1	1	2	3	3	4	4	5	1	2	3
hit		X	X	X	X	✓	X	X	✓	X	✓	X	X	X	X
v							1	2		3		4	5	1	2

- Total references: 14
- Total distinct references: 5
- Hits: 3
- Faults: 11
- **Hit rate:** 3/14 = **21.428571428571**%
- **Fault rate:** 11/14 = **78.571428571429**%

Τελευταία περιεχόμενα μνήμης 1 , 5, 4 , 3

Παρατηρώ πως το hit ratio παραμένει ίδιο και στις δυο περιπτώσεις.

c

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ref		1	2	3	4	1	5	1	3	2	4	3	4	5	1
f		1	2	3	4	1	5	1	3	2	4	3	4	5	1
f			1	2	3	4	1	5	1	3	2	4	3	4	5
f				1	2	3	4	4	5	1	3	2	2	3	4
hit		X	X	X	X	X	X	✓	X	X	X	✓	✓	X	X
v					1	2	3		4	5	1			2	3

Φεβρουάριος 2024

Θέμα 1ο (20 μονάδες)

Σε ένα σύστημα έρχονται τέσσερα jobs για εκτέλεση με τα παρακάτω χαρακτηριστικά:

Job	Χρόνος άφιξης	Χρόνος εκτέλεσης
1	0 sec	130 sec
2	40 sec	25 sec
3	70 sec	50 sec
4	120 sec	35 sec

Υπολογίστε το μέσο χρόνο επιστροφής (Average Turnaround Time), μέσο ανηγμένο χρόνο επιστροφής (Average Weighted Turnaround Time), μέσο χρόνο αναμονής (Average Waiting Time) και μέσο ανηγμένο χρόνο αναμονής (Average Weighted Waiting) αν το σύστημα χρησιμοποιεί τους αλγόριθμους Εκ Περιτροπής (Round Robin) και Συντομότερος Εναπομένων Χρόνος Πρώτα (SRTN: Shortest Remaining Time Next). Αγνοήστε τον χρόνο μεταγωγής διεργασιών (Task Switching Time).

Λύση:

Θέμα 1. Μονάδες 20

a) Round Robin

ATT = 135 AWTT = 2.48

AWT = 75 AWWT = 1.48

b) SRTN

ATT = 87.5 AWTT = 1.21

AWT = 27.5 AWWT = 0.21

Φεβρουάριος 2024

Θέμα 2ο (20 μονάδες)

Έστω ότι δίνεται μια επέκταση του προβλήματος του παραγωγού καταναλωτή με δύο παραγωγούς και ένα καταναλωτή που εκτελούνται ασύγχρονα και ταυτόχρονα. Το σύστημα έχει τους παρακάτω περιορισμούς:

1. Κάθε διεργασία έχει ένα κρίσιμο τμήμα που τοποθετεί ή εξάγει στοιχεία από έναν απομονωτή.
2. Ο καταναλωτής δεν μπορεί να εισέλθει στο κρίσιμο τμήμα του πριν εκτελεσθεί το γεγονός ενός τουλάχιστον παραγωγού. Επίσης, όταν διαβάζει από τον απομονωτή, γνωρίζει αν διάβασε μόνο του πρώτου παραγωγού, μόνο του δεύτερου ή και των δύο.
3. Κανένας από τους παραγωγούς δεν μπορεί να επανεισέλθει (loop) στο κρίσιμο τμήμα του πριν εκτελεσθεί το γεγονός του καταναλωτή. Εξαίρεση αποτελεί το πρώτο πέρασμα των παραγωγών.
4. Καμία διεργασία δεν μπορεί να γράψει ή να διαβάσει αν κάποια άλλη διεργασία γράφει ή διαβάζει.

```
Process Producer 1          Process Producer 2          Process Consumer 1
   :                           :                           :
   repeat                      repeat                      repeat
   :                           :                           :
   :                           :                           :
   critical section            critical section            critical section
   :                           :                           :
   :                           :                           :
until forever               until forever               until forever
```

Χρησιμοποιείστε semaphores για να επιτύχετε αμοιβαίο αποκλεισμό και συγχρονισμό των διεργασιών.

Λύση:

Θέμα 2. Μονάδες 20

```
Semaphore P1=1, P2=1, C=0
Process Producer 1          Process Producer 2          Process Consumer 1
    :                           :                           :
repeat                      repeat                      repeat
    :                           :                           :
   down(P1)                    down(P1)                    down(C)
   critical section            critical section            critical section
   up(C)                       up(C)                       if (read_from_1) then up(P1)
    :                           :                          if (read_from_2) then up(P2)
until forever               until forever                   :
                                                        until forever
```

Φεβρουάριος 2024

Θέμα 3ο (15 μονάδες)

Η παρακολούθηση του ελεύθερου χώρου στο δίσκο μπορεί να γίνει με τη χρήση μιας λίστας ελεύθερων μπλοκ ή ενός χάρτη ψηφίων. Οι διευθύνσεις του δίσκου απαιτούν Δ bits. Αν ένας δίσκος έχει χωρητικότητα M μπλοκ από τα οποία E είναι ελεύθερα, δώστε τη συνθήκη με την οποία η λίστα ελευθέρων μπλοκ χρησιμοποιεί λιγότερο χώρο από το χάρτη ψηφίων. Αν το Δ έχει τιμή 16 bits, εκφράστε την απάντησή σας ως ποσοστό επί τοις εκατό του χώρου δίσκου ο οποίος πρέπει να είναι ελεύθερος.

Λύση:

Θέμα 3. Μονάδες 15

ΕΔ<Μ

16Ε<Μ => Ε/Μ<1/16 = 6.25%

Φεβρουάριος 2024

Θέμα 4ο (15 μονάδες)

Σε ένα σύστημα που χρησιμοποιεί αποφυγή αδιεξόδου (deadlock avoidance) υπάρχουν δύο (2) εκτυπωτές και δύο (2) drives. Σε κάποια χρονική στιγμή της λειτουργίας του μπαίνουν τρείς διεργασίες για εκτέλεση. Οι διεργασίες έχουν ενημερώσει το λειτουργικό σύστημα ότι θα χρειαστούν τα εξής περιφερειακά:

Διεργασία	Εκτυπωτές	Disks
1	2	2
2	2	1
3	1	2

Όταν αρχίσουν να εκτελούνται, οι διεργασίες ζητούν περιφερειακά με την εξής σειρά:

1. Διεργασία 3 disk
2. Διεργασία 1 disk
3. Διεργασία 2 disk
4. Διεργασία 3 disk
5. Διεργασία 3 printer

Σε ποιές αιτήσεις θα αρνηθεί το λειτουργικό να δώσει το περιφερειακό; Είναι απαραίτητο να εξηγήσετε το γιατί, αναφερόμενοι στον αλγόριθμο που χρησιμοποιεί το λειτουργικό σύστημα.

Λύση:

Θέμα 4. Μονάδες 15

1. Διεργασία 3 disk **Ναι** (**ασφαλές**):

	R1	R2
*P1	0	0
*P2	0	0
*P3	0	1

	R1	R2
P1	2	2
P2	2	1
P3	1	1

R1	R2
2	1

2. Διεργασία 1 disk **Όχι** (ανασφαλές) :

	R1	R2
P1	0	1
P2	0	0
P3	0	1

	R1	R2
P1	2	1
P2	2	1
P3	1	1

R1	R2
2	0

3. Διεργασία 2 disk **Ναι** (ασφαλές):

	R1	R2
*P1	0	0
*P2	0	1
*P3	0	1

	R1	R2
P1	2	1
P2	2	0
P3	1	1

R1	R2
2	0

4. Διεργασία 3 disk **Όχι** (δεν έχει)
5. Διεργασία 3 printer **Όχι** (ανασφαλές):

	R1	R2
P1	0	0
P2	0	1
P3	1	1

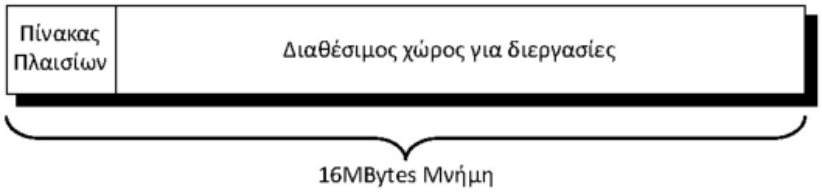
	R1	R2
P1	2	1
P2	2	0
P3	0	1

R1	R2
1	0

Φεβρουάριος 2024

Θέμα 5ο (15 μονάδες)

Ένα υπολογιστικό σύστημα διαθέτει την μνήμη που φαίνεται στο παρακάτω σχήμα για τις διεργασίες. Το λειτουργικό σύστημα διαχειρίζεται την μνήμη με πλαίσια σταθερού μεγέθους. Ένας πίνακας πλαισίων, που αποθηκεύεται στην μνήμη, περιέχει την κατάσταση κάθε πλαισίου της μνήμης. Πόσο μεγάλο θα πρέπει να είναι κάθε πλαίσιο;



Οι σχεδιαστές αποφάσισαν ότι το μέγεθος των πλαισίων μπορεί να είναι είτε 4K, είτε 16K, είτε 48K bytes. Ποιά από αυτές τις επιλογές θα ελαχιστοποιήσει τον χώρο που δεν χρησιμοποιείται από τις διεργασίες επειδή είτε καταλαμβάνεται από τον πίνακα πλαισίων είτε χάνεται λόγω κατακερματισμού; Υποθέστε ότι:

- Κατά μέσο όρο 8 διεργασίες βρίσκονται στη μνήμη. Ο μέσος χώρος που χάνεται λόγω κατακερματισμού είναι ½ πλαίσιο για κάθε διεργασία.
- Ο πίνακας πλαισίων έχει μια εγγραφή για κάθε πλαίσιο και κάθε εγγραφή χρειάζεται 40 bytes.

Λύση

Θέμα 5. Μονάδες 15

P: αριθμός πλαισίων, S: μέγεθος πλαισίου

$$P = \frac{16M - 40P}{S} \Longrightarrow P = \frac{16M}{S + 40}$$

α) S=4K

Χάνεται ½X4KX8 από κατακερματισμό + 40P από τον πίνακα.

Σύνολο ≈ **176K**

β) S=16K

Χάνεται ½X16KX8 από κατακερματισμό + 40P από τον πίνακα.

Σύνολο ≈ **104K**

α) S=48K

Χάνεται ½X48KX8 από κατακερματισμό + 40P από τον πίνακα.

Σύνολο ≈ **205K**

Συμφέρει το (β) **S=16K**

Φεβρουάριος 2024

Θέμα 6ο και τελευταίο (15 μονάδες)

Ας υποθέσουμε ότι ένας υπολογιστής μπορεί να διαβάσει ή να γράψει μια λέξη μνήμης σε 10 nsec. Ας υποθέσουμε επίσης ότι όταν προκύπτει μια διακοπή (interrupt), οι 32 καταχωρητές της CPU ο μετρητής προγράμματος και ο PSW (Processor Status Word) τοποθετούνται στη στοίβα. Ποιός είναι μέγιστος αριθμός διακοπών ανά δευτερόλεπτο που μπορεί να επεξεργαστεί αυτός ο υπολογιστής αν η εκτέλεση της ρουτίνας διακοπής απαιτεί 4μsec;

Λύση:

Θέμα 6. Μονάδες 15

Χρόνος κλήσης ISR = χρόνος εκτέλεσης + αποθήκευση καταχωρητών + ανάκληση καταχωρητών

t=4μs+2X(32+1+1)X10ns = 4680ns

Κλήσεις ανά δευτερόλεπτο= 1s/4680ns = **213,675 ≈ 213,000**