Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών
και Πληροφορικής

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Λειτουργικά Συστήματα
Ακαδημαϊκό Έτος 2018-2019
2η Άσκηση

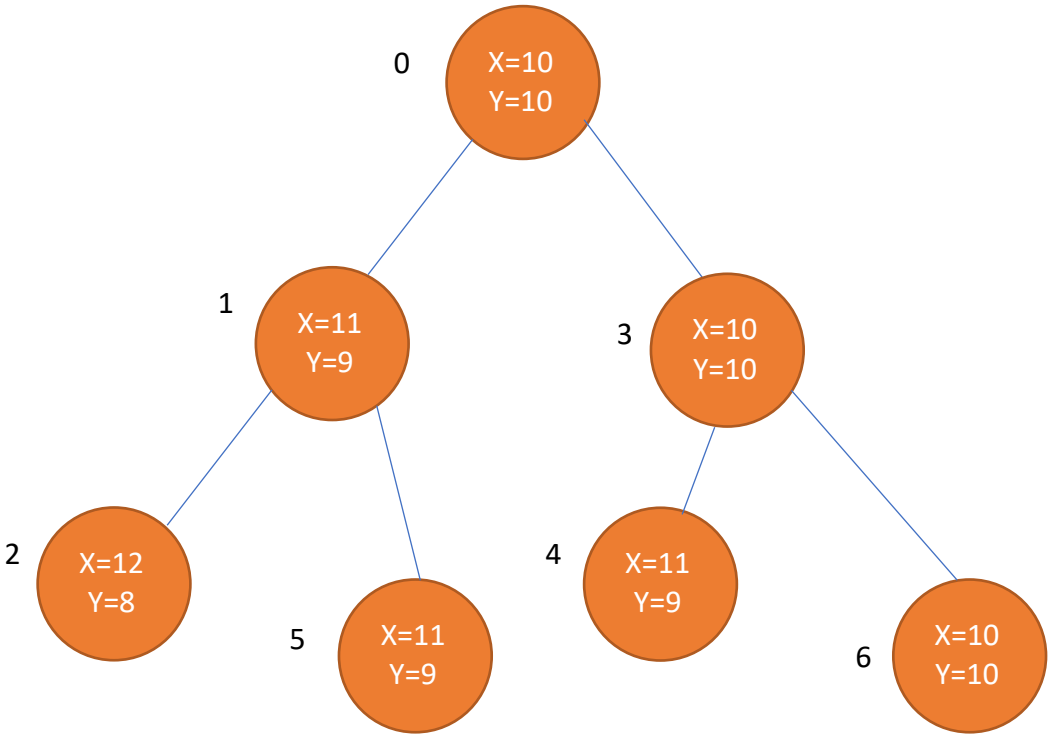
Όνομα:	ΑΜ:
Αγγελής Πέτρος	236268
Δημητρούκα Γιαννούλα	236291

Μέρος 1ο

Ερώτημα Α:
Στο πρόγραμμα που μας δίνεται, γίνεται χρήση της Fork() με την οποία δημιουργούνται διεργασίες(processes) με σχέσεις γονέα-παιδιού. Έτσι βάση τη θεωρία, στη Fork() τα αποτελέσματα που θα προκύψουν από τις printf() είναι τα εξής:

x = 11, y = 9.
x = 12, y = 8.
x = 10, y = 10.
x = 11, y = 9.
x = 11, y = 9.
x = 10, y = 10.

Επίσης ξέρουμε ότι η Fork παίρνει τη τιμή 0 για το παιδί και τη τιμή το pid του παιδιού για τον γονέα. Έτσι από τις If που υπάρχουν μέσα στο πρόγραμμα θα εξαρτάται ποιο θα είναι το αποτέλεσμα του x και y. Η σειρά με την οποία θα εκτελεστούν οι διεργασίες φαίνεται στο παρακάτω γράφημα:



Αναλυτικά:

Στην αρχή δίνονται οι τιμές του x & y.Αμέσως μετά γίνεται η πρώτη Fork() στην οποία δημιουργείται ο πρώτος γονέας με pid !=0 οπότε θα γίνουν οι πράξεις που δίνει το γράφονται στο κώδικα και θα εκτελεστεί η printf() μετά το πρώτο if.Έπειτα θα εκτελεστεί η δεύτερη Fork(), στην οποία πάλι ο πατέρας θα έχει pid !=0 και θα ξανά γίνουν οι πράξεις που γράφονται στον κώδικα. Θα εκτελεστεί η printf() που βρίσκεται κάτω από την δεύτερη Fork(), άρα έχουμε x=11 y=9, x=12 y=8. Στη συνέχεια θα εκτελεστεί το πρώτο παιδί, αλλά εφόσον το pid = 0, δεν θα γίνουν πράξεις και θα εκτελεστεί η printf() με τα αρχικά νούμερα που έχουν δοθεί στον κώδικα. Από την δεύτερη Fork() θα δημιουργηθεί ένας γονέας ο οποίος έχει pid != 0, θα γίνουν οι πράξεις μέσα στην if και θα εμφανιστεί το αποτέλεσμα. Μέχρι στιγμής έχουμε x=11 y=9, x=12 y=8, x=10 y=10,

x=11 y=9.Πλέον θα εκτελεστεί το παιδί του δέυτερου Fork() του γονέα του πρώτου Fork() και εφόσον το pid != 0,θα εκτελεστεί απλά η printf() η οποία θα βγάλει ίδια αποτελέσματα με τον πατέρα. Τέλος σειρά έχει στη δεύτερη Fork() το παιδί της από παιδί της πρώτης Fork() που το pid != 0 και έτσι θα εκτελεστεί η printf() απευθείας, δίνοντας τα αποτελέσματα του γονέα. Έτσι έχουμε x=11 y=9, x=12 y=8, x=10 y=10, x=11 y=9, x=11 y=9,x=10 y=10.

Ερώτημα Β:

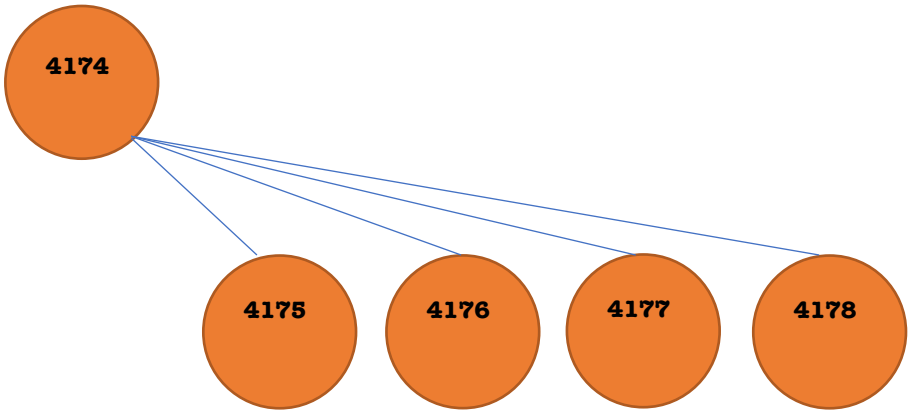
```
int main(void){
   int i, tmp;
   pid_t pid[4];

   for (i=0;i<4;i++) {
      pid[i] = fork();
      if (pid[i] == 0) {
         break;
      }
   }

   if (pid[0] != 0 && pid[1] != 0 && pid[2] != 0 && pid[3] != 0) {
      printf("I'm your father [pid: %d, ppid: %d]\n",getpid(),getppid());
      for(i=0;i<4;i++) {
         wait(&tmp);
      }
   } else {
      printf("NOOOOOOOOOOO [pid: %d, ppid: %d]\n",getpid(),getppid());
   }

   return 0;
}
```

Στο παραπάνω πρόγραμμα με την βοήθεια της for επιτυγχάνουμε να παράξουμε 4 παιδία ενώ έχουμε 1 γονέα από την fork.Στη κονσόλα θα εκτυπωθεί:

I'm your father [pid: 4174, ppid: 3866]
NOOOOOOOOOOO [pid: 4175, ppid: 4174]
NOOOOOOOOOOO [pid: 4176, ppid: 4174]
NOOOOOOOOOOO [pid: 4177, ppid: 4174]
NOOOOOOOOOOO [pid: 4178, ppid: 4174]

Παρατηρούμε ότι από το ίδιο pid του πατέρα δημιουργούντα τα 4 παιδιά

Το γράφημα των διεργασιών είναι:



Ερώτημα Γ:

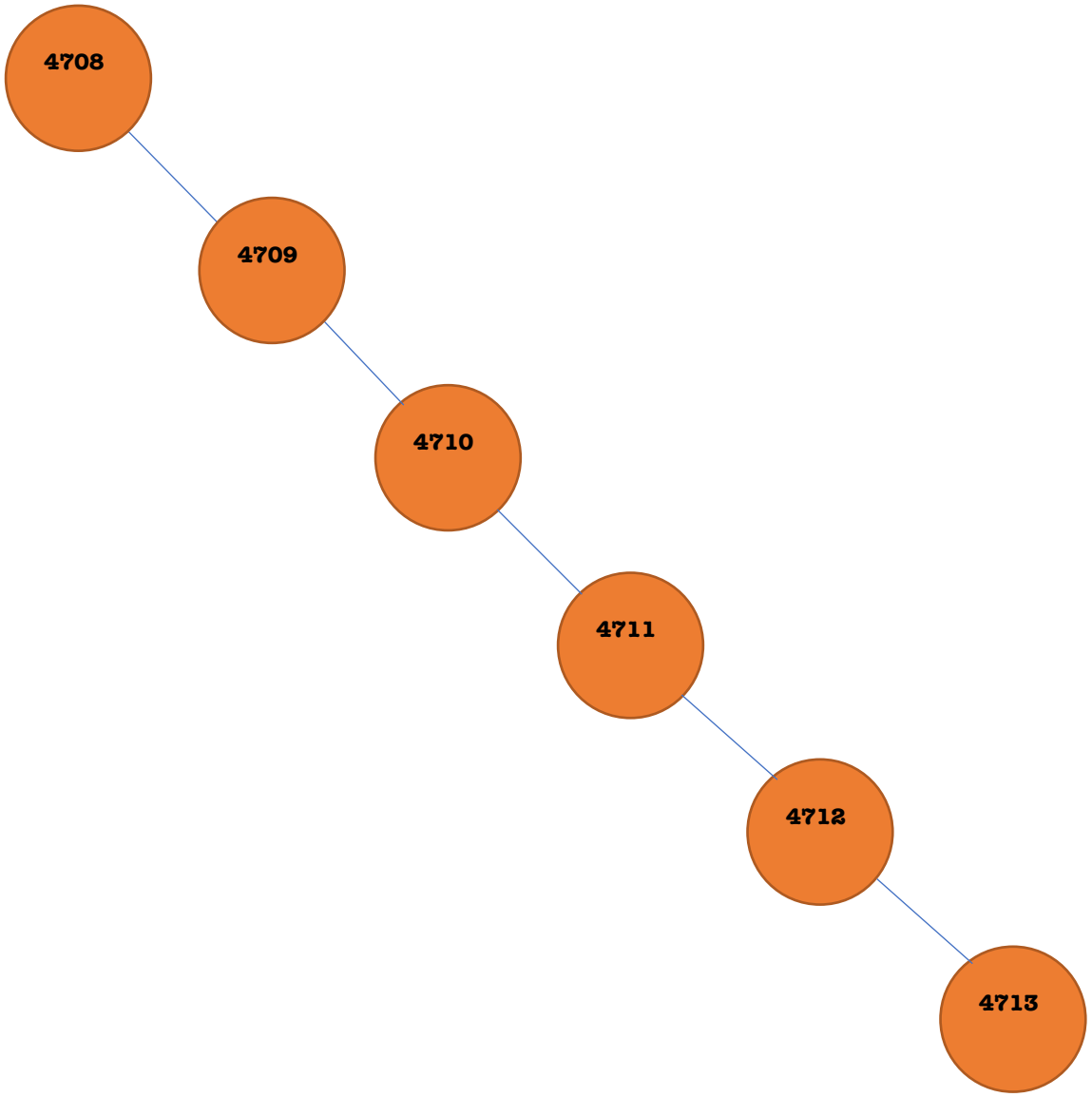
```
#define N   5

int main()
{
   int i;
   int pid;



   for (i=0; i<N; i++)
   {
      pid = fork();

      if (pid > 0)
      {
         printf(" Father = %5i, Id = %5i, Child = %5i\n", getppid(), getpid(), pid);
         wait(NULL);
         break;
      }
   }

   return (0);
}
```

Στο παραπάνω πρόγραμμα με την βοήθεια της for επιτυγχάνουμε να παράξουμε μια αλυσίδα η οποία θα πηγαίνει από γονέα σε παιδί .

Στη κονσόλα θα εκτυπωθεί:

```
Father = 3866, Id = 4708, Child = 4709
Father = 4708, Id = 4709, Child = 4710
Father = 4709, Id = 4710, Child = 4711
Father = 4710, Id = 4711, Child = 4712
Father = 4711, Id = 4712, Child = 4713
```

Το γράφημα των διεργασιών είναι:



Ερώτημα Δ:

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>
#include <time.h>

void nothing(void) {
   int x=0;
      x=x+1;
   //sleep(1); /* με το sleep δοκιμάζουμε εαν ο κώδικας δουλεύει οπως πρέπει καθώς ο μ.ο. κάθε διεργασίας είναι πολύ μικρός σε ένα σύγχρονο σύστημα*/
   return;
}

int main() {

   int x = 0;
   int i = 0, j = 0;
   time_t ft = 0;
   float mo;
   time_t begin,end;
   pid_t pids[100];
   pid_t curr_pid;
   int status;

   begin = time(NULL);
   printf("Αρχική τιμή δευτερολέπτων: %ld\n", begin);

   while (i < 100) {
      curr_pid = fork();
      nothing();

      if (curr_pid == 0)
         break;
      else
         pids[i] = curr_pid;

      i++;
   }

   if (curr_pid != 0) {
      for(i = 0; i < 100; i++){
         waitpid(pids[i], &status ,WEXITED);
      }
      end = time(NULL);
      ft = end - begin;

      mo = ft/100,0;

      printf("Τελική τιμή δευτερολέπτων %ld\n",end);
```

```
      printf("Μέσος όρος %f \n", mo);
   }


   return 0;
}
```

Στο παραπάνω κώδικα παίρνουμε τον αρχικό χρόνο στο begin από την time_t καθώς και τον τελικό στην end και ανάλογα με τον αριθμό των διεργασιών διαιρούμε και υπολογίζουμε τον μέσο όρο.

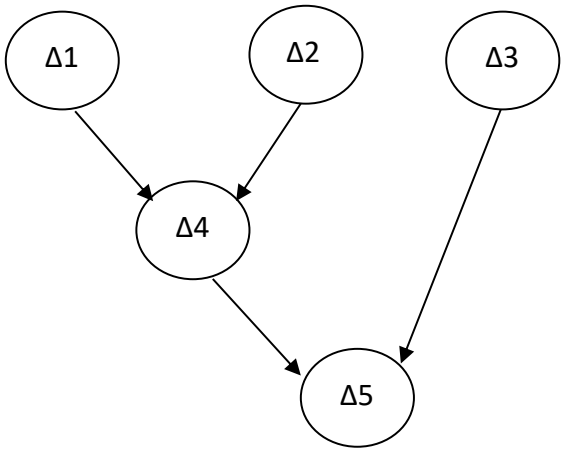
Ερώτημα Ε:

```
#include <sys/wait.h>
#include <unistd.h>
#include <stdio.h>        /* printf()              */
#include <stdlib.h>       /* exit(), malloc(), free() */
#include <sys/types.h>    /* key_t, sem_t, pid_t     */
#include <sys/shm.h>      /* shmat(), IPC_RMID       */
#include <errno.h>        /* errno, ECHILD           */
#include <semaphore.h>    /* sem_open(), sem_destroy(),
#include <fcntl.h>        /* sem_wait().. */



typedef sem_t Semaphore;
void display(char *str);

Semaphore *synch1;
Semaphore *synch2;

int  main(void) {
 int i, tmp;
   pid_t pid[5];
   int child_status;
   system("clear");

   for (i=0;i<5;i++) {
      pid[i] = fork();
      if (pid[i] == 0) {
         break;
      }
   }


   synch1 = sem_open ("Sem1", O_CREAT | O_EXCL, 0644, 0);
   synch2 = sem_open ("Sem2", O_CREAT | O_EXCL, 0644, 0);
/* δήλωση των σημαφόρων */
```

```
    /* 1ο παιδί */
  if (pid[0] == 0)
  {

     system ("cat file");
     sem_post (synch1);

  }
  /* 2ο παιδί */
  else if (pid[0] != 0 && pid[1] == 0)
  {

   system("ps -l");
   sem_post (synch1);
  }
  /* 3ο παιδί*/
  else if (pid[0] != 0 && pid[1] != 0 && pid[2] == 0)
  {

    system("ls -l");

   sem_post (synch2);
  }
  /* 4ο παιδί*/
  else if (pid[0] != 0 && pid[1] != 0 && pid[2] != 0 && pid[3] == 0)
  {
   sem_wait (synch1);
     sem_wait (synch1);
   system("awk");


  }
  /* 5ο παιδί*/
  else if (pid[0] != 0 && pid[1] != 0 && pid[2] != 0 && pid[3] != 0 && pid[4] == 0)
  {
   sem_wait (synch2);

    system("ps -l");


  }
 else
 {
for (i = 0; i < 5; i++)
{
   pid_t wpid = waitpid(pid[i], &child_status, 0);

   if (WIFEXITED(child_status))
        {

        }
        else
        {
```

```
            printf("Child: %d terminate abnormally\n", wpid);
          }
      }
      sem_unlink ("Sem1");            /* απελευθέρωση των σημαφόρων */
      sem_close(synch1);
      sem_unlink ("Sem2");
      sem_close(synch2);


   }


   exit (0);
}
```

Στο παραπάνω πρόγραμμα με βάση την προτεραιότητα του γράφου εκτελούνται οι εντολές συστήματος.



Ερώτημα Στ:

```
#include <sys/wait.h>
#include <time.h>
#include <unistd.h>
#include <string.h>
#include <stdio.h>        /* printf()              */
#include <stdlib.h>       /* exit(), malloc(), free() */
#include <sys/types.h>    /* key_t, sem_t, pid_t     */
#include <sys/shm.h>      /* shmat(), IPC_RMID       */
#include <errno.h>        /* errno, ECHILD           */
#include <semaphore.h>    /* sem_open(), sem_destroy(), sem_wait().. */
#include <fcntl.h>        /* O_CREAT, O_EXEC         */

#define  ONE     1
typedef sem_t     Semaphore;

Semaphore *mutex;             /*     synch semaphore     *//*shared */
int main ()
```

```
{

  int i;
  char w;
  key_t shmkey;              /* shared memory key       */
  int shmid;                 /* shared memory id        */
  pid_t pid;                 /* fork pid                */
  char *p;              /* shared variable         *//*shared */
  unsigned int n;            /* fork count              */
  unsigned int value;        /* semaphore value         */
      int *nSeconds;

      srand(time(NULL));
  shmkey = ftok ("/dev/null", 5);
  shmid = shmget (shmkey, sizeof (char), 0644 | IPC_CREAT);



      /* ελεγχος λαθους κοινης μνημης */
  if (shmid < 0)
  {
    perror ("shmget\n");
    exit (1);
  }

  p  = (char *) shmat (shmid, NULL, 0);

  printf ("Πόσα παιδιά χρειάζονται?\n");
  printf ("Αριθμός παιδιών: ");
  scanf  ("%u", &n);

  mutex = sem_open ("pSem", O_CREAT | O_EXCL, 0644, ONE);
  nSeconds = (int *)malloc(n * sizeof(int));

  for (i=0; i<n; i++) nSeconds[i] = rand()%4 + 1;

  /* fork child processes */
  for (i = 0; i < n; i++)
  {
    pid = fork ();

        if (pid < 0) {
    /* check for error      */
      sem_unlink ("pSem");
      sem_close(mutex);

      printf ("Fork error.\n");
    }
    else if (pid == 0) /* child processes */
      break;
  }

  if (pid > 0) /*γονέας*/
```

```
      {
     /* wait for all children to exit */
     while (pid = waitpid (-1, NULL, 0))
     {


       if (errno == ECHILD)
          break;
     }
      printf("%s ", p);
   printf("\n");
     /* shared memory detach */
     shmdt (p);
     shmctl (shmid, IPC_RMID, 0);

     sem_unlink ("pSem");
     sem_close(mutex);
  }
  else /*παιδί*/
      {
     sem_wait (mutex);        /* DOWN operation */
         printf ("Εκτέλεση %dου παιδιού\n", i+1);
     printf (" Περίμενε %d δευτερόλεπτα.\n", nSeconds[i]);
     w = *p;
     printf("Δωσε κείμενο χωρίς κενά: \n" );
     printf("Κείμενο:");
    scanf("%s",  &w);
     strcat(p, &w);
     sleep (nSeconds[i]);
      printf ("\n παιδί(%d) ξεφεύγει από τη κρίσιμη περιοχή\n", i+1);
     sem_post (mutex);        /* UP operation */
         }

  exit (0);

}
```

Στον παραπάνω κώδικα δηλώνεται από τον χρήστη ένας αριθμός παιδιών n και με βάση αυτό δημιουργούνται.Για κάθε παιδί δίνεται ένα μήνυμα στον χρήστη που του λέει να δώσει ένα κείμενο και εφόσον επαναληφθεί για όλα τα παιδιά εκτυπώνεται από τον πατέρα στο τέλος.

ΜΕΡΟΣ 2ο

Ερώτημα Α

Τμήμα	Βάση	Όριο
0	1024	1024
1	9896	128
2	512	128
3	3912	1400
4	1536	1024
5	5688	2000

α) Ο παραπάνω πίνακας δεν μπορεί να είναι πίνακας διεργασίας, γιατί το τμήμα 0 και 4 συμπίπτουν.

β)

(0, 256)	1024 + 256 = 1208
(1, 40)	9896 + 40 = 9936
(2, 512)	Εσφαλμένη αναφορά (illegal reference), παγίδευση στο λειτουργικό σύστημα: το μήκος του τμήματος 2 είναι 128 και η τιμή 512 το υπερβαίνει.
(3, 1000)	3912 + 1000 = 4912
(5, 1536)	5688 + 1536 = 7224

Ερώτημα Β

Λογική μνήμη	Πίνακας σελίδων	Φυσική	Πλαίσιο
Σελίδα 0	4		0
Σελίδα 1	7		1
Σελίδα 2	15	Δ1, Σ4	2
Σελίδα 3	12		3
Σελίδα 4	2	Δ1, Σ0	4
Σελίδα 5	10		5
Σελίδα 6	13		6
Σελίδα 7	18	Δ1, Σ1	7
		Δ2, Σ0	8
			9
		Δ1, Σ5	10
		Δ2, Σ3	11
		Δ1, Σ3	12
		Δ1, Σ6	13
		Δ2, Σ2	14
		Δ1, Σ2	15
		Δ3, Σ0	16
		Δ2, Σ7	17
		Δ1, Σ7	18
		Δ2, Σ6	19
		Δ2, Σ1	20
		Δ3, Σ1	21
		Δ2, Σ5	22
		Δ2, Σ4	23
		Δ3, Σ4	24
		Δ3, Σ7	25
		Δ3,Σ5	26
		Δ3,Σ3	27
			28
		Δ3, Σ6	29
		Δ3, Σ2	30
			31

Λογική μνήμη
Σελίδα 0
Σελίδα 1
Σελίδα 2
Σελίδα 3
Σελίδα 4
Σελίδα 5
Σελίδα 6
Σελίδα 7

Πίνακας σελίδων
8
20
14
11
23
22
19
17

Λογική μνήμη Διεργασίας 3
Σελίδα 0
Σελίδα 1
Σελίδα 2
Σελίδα 3
Σελίδα 4
Σελίδα 5
Σελίδα 6
Σελίδα 7

Πίνακας σελίδων
16
21
30
27
24
26
29
25

Ερώτημα Γ

Με δεδομένο ότι μια σελίδα στη λογική μνήμη έχει μέγεθος 8 Kbytes, μπορούμε να γράψουμε ότι είναι $2^3 * 2^{10} = 2^{13}$ bytes. Επίσης, η διεργασία 1 προσπελαύνει την λογική θέση μνήμης 30784_{10}, η οποία στο δυαδικό γράφεται: 0111100001000000. Επομένως έχουμε 16 bits που τοποθετούμε στη λογική διεύθυνση μνήμης.

Έχουμε 16-3=3. Τα υπόλοιπα 13 είναι η μετατόπιση. 011-**1100001000000**. Αφού τα τρία πρώτα bits μας οδηγούν στην τέταρτη θέση του πίνακα (Σελίδα 3) της λογικής μνήμης και αντίστοιχα στο 12, στο πίνακα σελίδων, δηλαδή το (Δ1,Σ3) της φυσικής μνήμης. Κάνουμε μετατροπή του 12 στο δυαδικό: $12_{10} \rightarrow 1100_2$, και τα τοποθετούμε στις πρώτες θέσεις της φυσικής διεύθυνσης μνήμης και τα 13 bits της μετατόπισης.

Λογική Διεύθυνση Μνήμης Διεργασίας 1

0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0				

Φυσική Διεύθυνση Μνήμης

0	1	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0		

Ερώτημα Δ

Η ακολουθία αναφοράς μιας διεργασίας:

3 5 8 1 8 7 5 1 4 2 8 2 7 3 6 4 6 5 3 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	3	3	3	3	3	7	7	7	7	2	2	2	2	2	2	4	4	4	4	7
1		5	5	5	5	5	5	5	5	5	8	8	8	8	6	6	6	6	6	6
2			8	8	8	8	8	8	4	4	4	4	4	3	3	3	3	3	3	3
3				1	1	1	1	1	1	1	1	1	7	7	7	7	7	5	5	5
Σφάλμα→	X	X	X	X		X			X	X	X		X	X	X	X		X		X