Πανεπιστήμιο Κρήτης, Τμήμα Επιστήμης Υπολογιστών 30 Νοεμβρίου 2009

Κατανεμημένος Υπολογισμός
Εαρινό Εξάμηνο Ακ. Έτους 2009-10
Διδάσκουσα: Παναγιώτα Φατούρου

Προγραμματιστικές Εργασίες

Προθεσμία Παράδοσης:
Υλοποίηση: 14 Ιανουαρίου 2010
Αναφορά: 18 Ιανουαρίου 2010
Παρουσίαση: 22 Ιανουαρίου 2010

Ο κώδικας θα πρέπει να σταλεί με e-mail στον βοηθό της εργασίας μέχρι τις 14 Ιανουαρίου 2010. Θα ακολουθήσει προφορική εξέταση και παρουσίαση των εργασιών και της βιβλιοθήκης που υλοποιήθηκε στην τάξη στις 22 Ιανουαρίου 2010.

Βοηθός Προγραμματιστικής Εργασίας: Ελευθέριος Κοσμάς (ekosmas@cs.uoi.gr)

Κάθε μια από τις εργασίες εμπεριέχει τη μελέτη κάποιων συστημάτων Transactional Memory, την υλοποίησή τους και την πειραματική τους μελέτη. Στη βιβλιογραφία έχουν προταθεί αρκετοί αλγόριθμοι STM. Στα πλαίσια της εργασίας αυτού του μαθήματος, εστιάζουμε στους πιο βασικούς από αυτούς. Συγκεκριμένα, κάθε εργασία θα μελετήσει, υλοποιήσει και παρουσιάσει δυο από τους αλγορίθμους που αναφέρονται παρακάτω.

[1] M. P. Herlihy, V. Luchangco, M. Moir and W. N. Scherer III. "Software Transactional Memory for Dynamic-Sized Data Structures", In *Proceedings of 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 92-101, July 2003.

[2] Fuad Tabba, Cong Wang, and Mark Moir, "NZTM: Nonblocking Zero-Indirection Transactional Memory", Proceedings of TRANSACT, 2007

(http://www.cs.rochester.edu/meetings/TRANSACT07/papers/tabba.pdf)

[3] Ελευθέριος Κοσμάς, «ΣΥΓΧΡΟΝΙΣΜΟΣ ΔΙΕΡΓΑΣΙΩΝ ΜΕΣΩ ΔΟΣΟΛΗΨΙΩΝ», Μεταπτυχιακή Εργασία Εξειδίκευσης, 2008.

[4] Michael F. Spear, Maged M. Michael and Christoph von Praun, "*RingSTM*: Scalable Transactions with a Single Atomic Instruction", *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, 2008 (http://www.cs.rochester.edu/u/spear/spaa08.pdf)

[5] Keir Fraser, "*Practical lock freedom"*, PhD thesis, Cambridge University Computer Laboratory (also available as *Technical Report UCAM-CL-TR-579*).

[6] D. Dice, O. Shalev and N. Shavit. "Transactional Locking II", In *Proceedings of the 20th International Symposium on Distributed Computing (DISC)*, September 2006.

Α. Εισαγωγή και περιγραφή αποτελεσμάτων εργασίας

Για την πλήρη εκμετάλλευση των πολυπύρηνων επεξεργαστών είναι απαραίτητο να απλοποιηθεί η διαδικασία παράλληλου προγραμματισμού. Μια νέα τεχνική παράλληλου προγραμματισμού που αναγνωρίζεται από τους ερευνητές ως η επικρατέστερη εναλλακτική μέθοδος παράλληλου προγραμματισμού είναι η Software Transactional Μνήμη (STM). Η STM υποστηρίζει την εκτέλεση *δοσοληψιών* από τις διεργασίες, οι οποίες περιέχουν λειτουργίες που ο χρήστης επιθυμεί να εκτελέσει στα διαμοιραζόμενα αντικείμενα. Η STM εγγυάται την ατομική εκτέλεση των λειτουργιών αυτών.

Σκοπός της παρούσας εργασίας είναι η μελέτη, υλοποίηση και πειραματική ανάλυση υπαρχόντων συστημάτων STM. Σε κάθε μια από τις εργασίες θα πρέπει να μελετηθούν οι αντίστοιχοι αλγόριθμοι και αρχικά να γραφεί ο κώδικάς τους υπό την μορφή ψευδοκώδικα (σε όσους δεν παρουσιάζεται ήδη). Θα πρέπει επίσης να επισημανθούν σημεία της περιγραφής ενός αλγορίθμου τα οποία δεν είναι ξεκάθαρα, καθώς και να αναφερθούν τυχόν υποθέσεις ή τροποποιήσεις που θεωρείτε απαραίτητες.

Έπειτα, θα πρέπει να υλοποιηθεί ο κάθε αλγόριθμος στη γλώσσα προγραμματισμού C χρησιμοποιώντας επιπρόσθετα μια βιβλιοθήκη η οποία παρέχει υλοποιήσεις των ατομικών λειτουργιών των ισχυρών καταχωρητών (τύπου CAS και LL/SC) που είναι απαραίτητοι για την υλοποίηση των STM αλγορίθμων.

Ο κάθε αλγόριθμος θα υλοποιηθεί υπό τη μορφή μιας βιβλιοθήκης συναρτήσεων και διαδικασιών, τις οποίες θα μπορεί ο χρήστης να καλεί για την επίτευξη εκτέλεσης κώδικα με ατομικό τρόπο κατά τη συγγραφή παράλληλου κώδικα. Η διεπαφή (interface) της βιβλιοθήκης αυτής παρουσιάζεται αναλυτικά στη συνέχεια. Η συγγραφή κώδικα για τη δημιουργία της βιβλιοθήκης αυτής είναι μια από τις βασικές εργασίες που ζητείται να εκτελέσετε.

Τέλος, η απόδοση των συγκεκριμένων αλγορίθμων θα πρέπει να μελετηθεί αναλυτικά. Μετρικά τα οποία θα μπορούσαν να μελετηθούν είναι το πλήθος (ή ποσοστό) των επιτυχημένων και αποτυχημένων δοσοληψιών, ο μέσος χρόνος ολοκλήρωσης μιας δοσοληψίας (που πιθανώς επανεκτελείται εάν αποτύχει) ως επιτυχημένης, ο αντίστοιχος αριθμητικός μέσος, το μέσο πλήθος αποτυχιών ανά δοσοληψία πριν αυτή καταφέρει τελικά να ολοκληρωθεί και άλλα.

Β. Απαραίτητοι Ορισμοί – Περιγραφή Διεπαφής Βιβλιοθήκης STM

Η STM αποτελείται από ένα σύνολο κελιών μνήμης, τα οποία ονομάζονται *transactional μεταβλητές*, ή *t-μεταβλητές* εν συντομία, και τα οποία επιτρέπεται να προσπελάζονται από τις διεργασίες μόνο μέσω των λειτουργιών που η STM παρέχει. Συγκεκριμένα η STM επιτρέπει στις διεργασίες να επικοινωνούν διαβάζοντας και ενημερώνοντας t-μεταβλητές με τη βοήθεια *δοσοληψιών*. Αυτό σημαίνει ότι οι διεργασίες περικλείουν τις λειτουργίες που επιθυμούν να εκτελέσουν στις t-μεταβλητές σε μια δοσοληψία και ζητούν από την STM να εκτελέσει τη συγκεκριμένη δοσοληψία. Εάν το πλήθος των t-μεταβλητών είναι σταθερό τότε η STM λέγεται *στατική*, ενώ εάν μπορεί να τροποποιηθεί (καλώντας κάποια συνάρτηση δημιουργίας νέων t-μεταβλητών) λέμε ότι χρησιμοποιείται *δυναμική STM*. Σημειώνεται ότι για την εκτέλεση μιας δοσοληψίας που χρησιμοποιεί δυναμικά δεδομένα (π.χ., που θέλει να εισάγει ένα νέο στοιχείο σε μια δυναμική δομή δεδομένων) απαιτείται η χρήση δυναμικής STM. Στην παρούσα εργασία θα ασχοληθούμε μόνο με δυναμικές δοσοληψίες.

Η STM μπορεί να μοντελοποιηθεί ως ένα ατομικό αντικείμενο που παρέχει κάποιες λειτουργίες στις διεργασίες που επιθυμούν να το χρησιμοποιήσουν. Οι λειτουργίες αυτές αποτελούν τη βιβλιοθήκη συναρτήσεων και διαδικασιών που προαναφέρθηκε και τις οποίες μπορεί να καλεί ο χρήστης.

Πιο συγκεκριμένα, μια μνήμη STM υποστηρίζει για κάθε διεργασία p_i τις εξής λειτουργίες:

i) *pointer BeginTransaction()*: Η λειτουργία αυτή χρησιμοποιείται από την p_i για να εκκινήσει μια νέα δοσοληψία. Η BeginTransaction επιστρέφει ένα δείκτη σε μια κατάλληλη δομή (διαφορετική για κάθε αλγόριθμο) που περιγράφει τη δοσοληψία που μόλις εκκινήθηκε. Ο χρήστης απαιτείται να χρησιμοποιήσει ως παράμετρο ένα τέτοιο δείκτη κατά την κλήση των υπολοίπων λειτουργιών που θα εκτελεστούν από αυτή τη δοσοληψία.

ii) *pointer CreateNewTmVar(pointer t, d)*: Η λειτουργία αυτή χρησιμοποιείται από την p_i που εκτελεί τη δοσοληψία που περιγράφεται από τον δείκτη t (ο οποίος επιστράφηκε από την BeginTransaction() κατά τη δημιουργία της δοσοληψίας αυτής). Σκοπός της είναι να δημιουργήσει μια νέα t-μεταβλητή tvar τα δεδομένα της οποίας αρχικοποιούνται ώστε να είναι ίδια με τα δεδομένα *d* (τα οποία σε αυτή την εργασία είναι ενός συγκεκριμένου τύπου που θα σας δοθεί). Η χρησιμότητα της λειτουργίας αυτής εξηγείται στη συνέχεια. Η CreateNewTmVar επιστρέφει στην p_i ένα δείκτη σε μια κατάλληλη δομή που περιγράφει την tvar. Ο χρήστης απαιτείται να χρησιμοποιήσει ως παράμετρο ένα τέτοιο δείκτη στην κλήση των λειτουργιών ReadTmVar και WriteTmVar που διαβάζουν και τροποποιούν αντίστοιχα την tvar.

   Κάθε υλοποίηση μνήμης STM αντιστοιχεί κάποιες επιπλέον πληροφορίες (metadata) σε κάθε t-μεταβλητή η οποία περιγράφει δεδομένα του χρήστη τα οποία πρόκειται να προσπελαστούν μέσω δοσοληψιών. Αυτές οι επιπλέον πληροφορίες αποτελούν την *αναπαράσταση της t-μεταβλητής* στο σύστημα STM. Μέσω της CreateNewTmVar ο χρήστης ενημερώνει το σύστημα STM για την ύπαρξη κάποιων ακόμα δεδομένων που πρόκειται να προσπελάζονται μέσω δοσοληψιών, έτσι ώστε να δημιουργηθεί μια νέα t-μεταβλητή που θα αντιστοιχισθεί στα δεδομένα αυτά και να αρχικοποιηθεί κατάλληλα ο τρόπος αναπαράστασής της στο εκάστοτε σύστημα STM.

iii) *(Boolean, d) ReadTmVar(t, tVar)*: Η λειτουργία αυτή χρησιμοποιείται από την p_i που εκτελεί τη δοσοληψία *t* (ο pointer που επιστράφηκε από την BeginTransaction) για να διαβάσει τα δεδομένα της t-μεταβλητής *tVar* (ο pointer που επιστράφηκε από την CreateNewTmVar). Η ReadTmVar επιστρέφει στην p_i δύο τιμές. Η πρώτη είναι μία τιμή Boolean η οποία είναι TRUE εάν η λειτουργία ReadTmVar ολοκληρώθηκε επιτυχώς και FALSE σε αντίθετη περίπτωση. Εάν επιστραφεί TRUE, τότε στην δεύτερη μεταβλητή επιστρέφονται τα δεδομένα *d* της tVar. Η *ReadTmVar* αποτυγχάνει αν π.χ., επιχειρεί να διαβάσει μη-συνεπή δεδομένα.

iv) *Boolean WriteTmVar (t, tVar, d)*: Η λειτουργία αυτή χρησιμοποιείται από την p_i που εκτελεί τη δοσοληψία *t* για να ενημερώσει τα δεδομένα της t-μεταβλητής *tVar* με την τιμή *d*. Η WriteTmVar επιστρέφει στην p_i μία τιμή Boolean η οποία είναι TRUE εάν η λειτουργία WriteTmVar ολοκληρώθηκε επιτυχώς και FALSE σε αντίθετη περίπτωση.

v) *Boolean CommitTransaction(t)*: Με τη λειτουργία αυτή η p_i δηλώνει ότι ολοκλήρωσε την εκτέλεση της δοσοληψίας *t* και ζητά από την STM να τερματίσει την *t* επιτυχώς. Η λειτουργία αυτή επιστρέφει TRUE σε περίπτωση επιτυχούς τερματισμού της t ως επιτυχημένης ή FALSE σε περίπτωση τερματισμού της t ως μη-επιτυχημένης.

vi) *Void AbortTransaction(t)*: Με τη λειτουργία αυτή η p_i δηλώνει ότι επιθυμεί τον τερματισμό της δοσοληψίας *t* ως μη-επιτυχημένη. Η λειτουργία αυτή εξασφαλίζει ότι η t θα τερματίσει μη-επιτυχώς και έτσι δεν επιστρέφει τίποτε.

Ως παράδειγμα, στο Σχήμα 1 παρουσιάζεται η ατομική λειτουργία αύξησης ενός μετρητή counter, με βάση το μοντέλο STM.

```
1. void AtomicCounterIncrement (TmVar counter) {
2.     int tmp;
3.     TmVar newTmVar;
4.     boolean bool;
```

```
5.    void *t;
6.    while (1) {
7.       t = BeginTransaction ();     // δημιουργία μιας νέας δοσοληψίας
8.       (bool, tmp) = ReadTmVar (t, counter);  // ανάγνωση της t-μεταβλητής counter
                                                // tmp = τιμή της t-μετ/τής counter
9.       tmp ++;
10.      bool = WriteTmVar (t, counter, tmp);   // εγγραφή στην t-μετ/τής counter
                                                // counter = tmp
11.      if (bool == false) {                   // αν η αυξημένη τιμή είναι απαρχαιωμένη
                                                // η WriteTmVar αποτυγχάνει
                                                // επιστρέφοντας FALSE
12.           AbortTransaction(t);
13.           continue;
14.      }
15.      if (CommitTransaction(t) == TRUE)
16.         break;                              // η δοσοληψία τερματίζει επιτυχώς
17.   }
18. }
```

Σχήμα 1: Αύξηση μετρητή με χρήση του βασικού μοντέλου STM.

Στο παραπάνω απλό παράδειγμα, η t-μεταβλητή είναι ένας ακέραιος και η μνήμη STM είναι στατική. Στην εργασία που θα υλοποιήσετε, η μνήμη STM είναι δυναμική (όπως προαναφέρθηκε). Ωστόσο, υποθέτουμε ότι κάθε διεργασία μπορεί να αποθηκεύει ως δεδομένα των t-μεταβλητών μόνο δείκτες σε αντικείμενα της δομής τύπου Object που περιγράφεται παρακάτω και αποτελεί τον κόμβο μιας λίστας ή ουράς:

```
typedef void * TmVar;

struct Object {
      int x;
      TmVar next;
}
```

Γ. Απαραίτητες Συμβάσεις

Στην παρούσα παράγραφο περιγράφεται ο τρόπος με τον οποίο ο χρήστης χρησιμοποιεί το μοντέλο STM. Μια δοσοληψία εκκινείται με την εκτέλεση της λειτουργίας BeginTransaction, η οποία επιστρέφει ένα δείκτη σε μια κατάλληλη δομή που περιγράφει τη δοσοληψία που μόλις εκκινήθηκε. Ο χρήστης χρησιμοποιεί το δείκτη αυτό σε όλες τις λειτουργίες που θέλει να εκτελέσει μέσω τις συγκεκριμένης δοσοληψίας. Από το σημείο αυτό και έπειτα η δοσοληψία που εκτελεί ο χρήστης είναι εκκρεμής έως ότου εκτελεστεί από το χρήστη είτε η λειτουργία CommitTransaction (για την ολοκλήρωση της δοσοληψίας ως επιτυχημένης) είτε η λειτουργία AbortTransaction (για την ολοκλήρωση της δοσοληψίας ως αποτυχημένης). Σημειώνεται ότι ο χρήστης μπορεί να έχει το πολύ μία εκκρεμή δοσοληψία κάθε χρονική στιγμή, δηλαδή κάθε κλήση της λειτουργίας BeginTransaction πρέπει να ακολουθείται από μια κλήση μιας εκ των λειτουργιών CommitTransaction ή AbortTransaction, χωρίς να παρεμβάλλεται στο ενδιάμεσο κάποια άλλη κλήση της BeginTransaction.

Ο χρήστης προσπελάζει τις t-μεταβλητές που επιθυμεί ή δημιουργεί κάποια νέα t-μεταβλητή μέσω μιας δοσοληψίας μόνο ενόσω η δοσοληψία αυτή είναι εκκρεμής, και μέσω των αντίστοιχων λειτουργιών

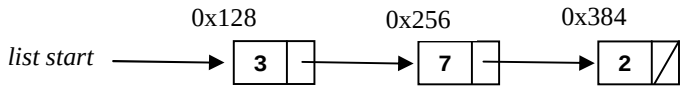
ReadTmVar, WriteTmVar και CreateNewTmVar. Ο χρήστης είναι ελεύθερος να καλεί τις λειτουργίες προσπέλασης για την ίδια t-μεταβλητή όσες φορές επιθυμεί. Συγκεκριμένα, την πρώτη φορά που διαβάζει τα δεδομένα μιας t-μεταβλητής τα προσπελάζει με την ReadTmVar. Κάθε φορά που ο χρήστης ενημερώνει κάποια t-μεταβλητή εκτελεί τη λειτουργία WriteTmVar.

Όταν ο χρήστης επιθυμεί να τελειώσει την εκτέλεση μιας δοσοληψίας καλεί είτε τη λειτουργία CommitTransaction για να επιχειρήσει να τερματίσει τη δοσοληψία ως επιτυχημένη ή τη λειτουργία AbortTransaction για να την τερματίσει ως μη-επιτυχημένη. Εάν οποιαδήποτε λειτουργία προσπέλασης επιστρέψει στο χρήστη για την Boolean μεταβλητή την τιμή FALSE, δηλώνοντας ότι η συγκεκριμένη λειτουργία απέτυχε, ο χρήστης καλεί άμεσα μία εκ των λειτουργιών CommitTransaction ή AbortTransaction ώστε να τερματίσει τη δοσοληψία που εκτελεί. Σημειώνεται ότι στην περίπτωση αυτή είναι βέβαιο, ανεξάρτητα με τη λειτουργία τερματισμού που καλείται, ότι η δοσοληψία θα τερματίσει ως μη-επιτυχημένη.

Ο χρήστης χρησιμοποιεί τη λειτουργία CreateNewTmVar, κάθε φορά που επιθυμεί να δημιουργήσει και να αρχικοποιήσει μια νέα t-μεταβλητή κατά την εκτέλεση μιας δοσοληψίας. Για παράδειγμα, αν ο χρήστης επιθυμεί να υλοποιήσει μια συνδεδεμένη λίστα βάσει κάποιου ατομικού αντικειμένου STM πρέπει να αντιστοιχίσει μια t-μεταβλητή σε κάθε κόμβο της λίστας, η οποία αποθηκεύει ένα δείκτη στον κόμβο αυτό. Στην περίπτωση αυτή, εάν ο χρήστης θέλει να διατηρήσει έναν δείκτη p από κάποιο κόμβο x προς κάποιο κόμβο y της λίστας, ο p δεν επιτρέπεται να δείχνει απευθείας στο y, αλλά πρέπει να δείχνει στην t-μεταβλητή που περιγράφει τον κόμβο y της λίστας. Αυτό σημαίνει ότι το μοντέλο STM δεν επιτρέπει στον χρήστη να διατηρεί δείκτες μεταξύ των ίδιων των διαμοιραζόμενων δεδομένων, αλλά ένα διαμοιραζόμενο δεδομένο x μπορεί να περιέχει δείκτη σε ένα διαμοιραζόμενο δεδομένο y μόνο εάν ο δείκτης δείχνει στην αντίστοιχη t-μεταβλητή που περιγράφει το y. Για το λόγο αυτό, ο δείκτης next του struct Object είναι τύπου void.

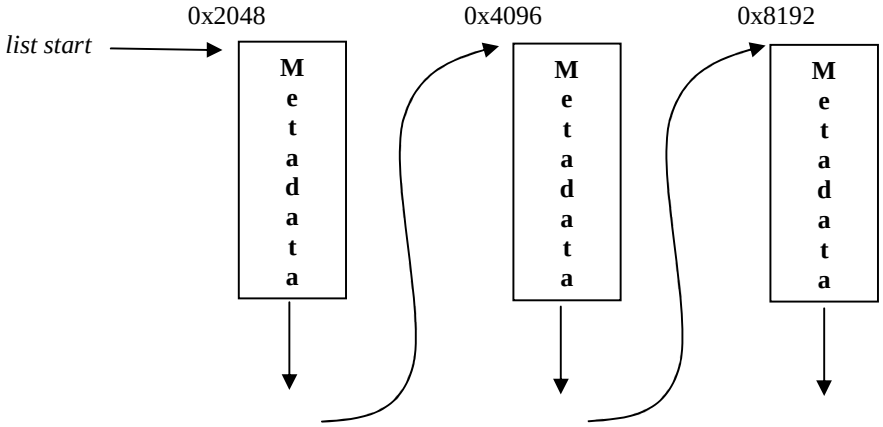
Παράδειγμα

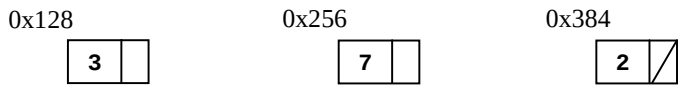
Έστω η λίστα του Σχήματος 2.



Σχήμα 2: Συνδεδεμένη λίστα.

Στο Σχήμα 3 παρουσιάζεται η μορφή της ίδια λίστας σε κάποιο σύστημα μνήμης STM, στο οποίο η αναπαράσταση μιας t-μεταβλητής συμβολίζεται αφαιρετικά με έναν κόμβο metadata. Παρατηρούμε πως η t-μεταβλητή του κόμβου 3 της λίστας είναι ένας δείκτης ο οποίος έχει την τιμή 0x248, δηλαδή την τιμή της διεύθυνσης στην οποία ξεκινά ο κόμβος metadata.





Σχήμα 3: Συνδεδεμένη λίστα σε μια μνήμη STM.

Στο Σχήμα 4 δίνεται ως παράδειγμα ο κώδικας της λειτουργίας εισαγωγής ενός κόμβου στη λίστα του Σχήματος 3. Υποθέτουμε πως κάθε κόμβος εισάγεται στο τέλος της λίστας. Επίσης υποθέτουμε πώς υπάρχει ένας κόμβος φρουρός στην αρχή της λίστας.

```
1.  void Insert (int num, TmVar listStart) {
2.      Object *tmpNode=null, *newNode;
3.      TmVar tmpNodeTmVar, newTmVar;
4.      boolean bool;
5.      void *t;
6.      while (1) {
7.          t = BeginTransaction ();           // δημιουργία μιας νέας δοσοληψίας
8.          tmpNodeTmVar = listStart;          // Εύρεση του τελευταίου στοιχείου της
                                               // λίστας. Μετά το do..while στην
                                               // tmpNode θα βρίσκονται τα δεδομένα του
                                               // στοιχείου αυτού και στην tmpNodeTmVar
                                               // η t-μεταβλητή που το περιγράφει.
9.          do {
10.             if (tmpNode != null)
11.                 tmpNodeTmVar = tmpNode->next;
12.             (bool, tmpNode) = ReadTmVar (t, tmpNodeTmVar);
13.             if (bool == false) break;
14.         } while (tmpNode->next == null);

15.         if (bool == false)
16.             AbortTransaction(t);
17.         else {
18.             newNode = (Node *)malloc (sizeof(Node));     // Αρχικοποίηση δεδομένων
                                                             // νέου στοιχείου
19.             newNode->num = num;
20.             newNode->next = null;
21.             newTmVar = CreateNewTmVar (t, newNode);      // Αρχικοποίηση μιας t-
                                                             // μεταβλητής που θα
                                                             // περιγράφει το νέο
                                                             // στοιχείο
22.             tmpNode->next = newTmVar;
23.             bool = WriteTmVar (t, tmpNodeTmVar, tmpNode);// Ενημέρωση τελευταίου
                                                             // κόμβου της λίστας
24.             if (bool == false)
25.                 continue;
26.             if (CommitTransaction(t) == TRUE)
27.                 break;
28.         }  // else
29.     }    // while
30. }  // Insert
```

Σχήμα 4: Λειτουργία εισαγωγής ενός κόμβου στη λίστα στου Σχήματος 3.

Δ. Παραδοτέα

Στο πρώτο μέρος της εργασίας, σας ζητείται με βάση το μοντέλο STM και το παράδειγμα της παραγράφου Γ να υλοποιήσετε τις λειτουργίες των δομών δεδομένων που θα χρησιμοποιήσετε στα πειράματά σας, π.χ. λίστα, ταξινομημένη λίστα, ουρά, κτλ. Επίσης, στο στάδιο αυτό πρέπει να παρουσιάσετε ενδεικτικό κώδικα των πειραμάτων που πρόκειται να εκτελεστούν. Τέλος στο στάδιο αυτό ζητείται να παραδώσετε την πρώτη έκδοση της αναφοράς της εργασίας όπου θα περιγράφετε τα πειράματα που πρόκειται να εκτελέσετε, τον αλγόριθμο που έχετε να υλοποιήσετε και τον σχεδιασμό της βιβλιοθήκης που θα προγραμματίσετε βάσει των αλγορίθμων που έχετε αναλάβει να μελετήσετε.

Στο δεύτερο στάδιο της εργασίας θα παραδώσετε μια βιβλιοθήκη συναρτήσεων και διαδικασιών που θα υλοποιούν δυο από τους αλγορίθμους STM που αναφέρθηκαν στην εισαγωγή.