Ατομική Διπλωματική Εργασία

ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΠΑΡΑΛΛΗΛΩΝ ΕΥΡΩΣΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ΤΟ ΕΡΓΑΛΕΙΟ JAVA SWING

Ελένη Σκιττίδου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2011

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΓΡΑΦΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΠΑΡΑΛΛΗΛΩΝ ΕΥΡΩΣΤΩΝ ΑΛΓΟΡΙΘΜΩΝ ΜΕ ΤΟ ΕΡΓΑΛΕΙΟ JAVA SWING

Ελένη Σκιττίδου

Επιβλέπων Καθηγητής
Χρύσης Γεωργίου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2011

Ευχαριστίες

Με το πέρας της διπλωματικής μου εργασίας θα ήθελα να ευχαριστήσω θερμά το δόκτορα Χρύση Γεωργίου, που είχε την ευθύνη για την επίβλεψή της και που μου έδωσε την ευκαιρία να εργαστώ πάνω στο συγκεκριμένο αντικείμενο. Επίσης, θέλω να τον ευχαριστήσω για την καθοδήγηση, τη βοήθεια και τις συμβουλές που μου πρόσφερε κατά την εκπόνηση της Ατομικής Διπλωματικής Εργασίας μου, που ήταν χρήσιμες και καθοριστικές για την ολοκλήρωσή της.

Περίληψη

Το γενικό θέμα αυτής της διπλωματικής είναι ο παραλληλισμός και οι παράλληλοι αλγόριθμοι. Δηλαδή η συνεταιρική και ταυτόχρονη επεξεργασία δεδομένων από περισσότερους από ένα επεξεργαστές, που αποσκοπεί στη γρήγορη επίλυση σύνθετων υπολογιστικών προβλημάτων. Διανύουμε μία εποχή όπου ο παραλληλισμός αποδεικνύεται ότι έχει σημαντικά πλεονεκτήματα σε σχέση με τον σειριακό υπολογισμό και βρίσκεται σε ακμή. Αυτός είναι ο λόγος που αποφασίστηκε το θέμα αυτής της διπλωματικής εργασίας να είναι η μελέτη και η γραφική αναπαράσταση κάποιων κύριων παράλληλων αλγορίθμων. Η αναπαράσταση της λειτουργίας σύνθετων παράλληλων αλγορίθμων κάνει πιο αποτελεσματική τη διδασκαλία και εκμάθηση τους.

Συνεπώς, σκοπός αυτής της εργασίας ήταν η δημιουργία γραφικής διεπαφής με τον χρήστη (GUI) και η γραφική αναπαράσταση παράλληλων εύρωστων αλγορίθμων με την χρήση της γλώσσας προγραμματισμού Java και του εργαλείου Java Swing σε συνδυασμό με τη κλάση Graphics της Java.

Στα αρχικά στάδια έγινε η εγκατάσταση του NetBeans, η μελέτη των παράλληλων αλγορίθμων, όπως και η μελέτη και η εξοικείωση με την γλώσσα προγραμματισμού Java, το εργαλείο Java Swing και τη κλάση Graphics της Java. Έπειτα έγινε η γραφική αναπαράσταση των παράλληλων και εύρωστων αλγορίθμων.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Σκοπός και Κίνητρο Διπλωματικές Εργασίας

Ο παραλληλισμός και οι παράλληλοι αλγόριθμοι προσφέρουν πολλά πλεονεκτήματα σε σχέση με το σειριακό υπολογισμό. Το κυριότερο πλεονέκτημά τους είναι ότι μπορούν να διεκπεραιώσουν πολύπλοκους και μεγάλους υπολογισμούς, σε αντίθεση με ένα σειριακό υπολογιστή. Ο τρόπος σκέψης και υλοποίησής τους όμως, καθώς και ο τρόπος λειτουργίας τους είναι πολύ διαφορετικός από το σειριακό. Επομένως, η διπλωματική εργασία που έχω αναλάβει έχει ως σκοπό να βοηθήσει στη διδασκαλία και την εκμάθηση της λειτουργίας κάποιων βασικών παράλληλων και εύρωστων αλγορίθμων με τη γραφική αναπαράσταση κάθε βήματος του αλγορίθμου.

Πιο συγκεκριμένα, έχει ως σκοπό τη δημιουργία γραφικής διεπαφής με το χρήστη (GUI) και τη γραφική αναπαράσταση παράλληλων εύρωστων αλγορίθμων. Η υλοποίηση των αλγορίθμων έγινε με τη χρήση της γλώσσας προγραμματισμού Java και του εργαλείου Java Swing σε συνδυασμό με τη κλάση Graphics της Java. Δηλαδή, θα γίνεται η γραφική αναπαράσταση των αλγορίθμων, βήμα προς βήμα, δίνοντας τη δυνατότητα στο χρήστη να παρέμβει προσθέτοντας τις δικές του επιλογές στον κάθε αλγόριθμο, όπως για παράδειγμα τον αριθμό των επεξεργαστών.

1.2 Μεθοδολογία Υλοποίησης

Η μεθοδολογία που ακολουθήθηκε για την υλοποίηση της παρούσας διπλωματικής εργασίας είναι η ακόλουθη: Αρχικά, έγινε η μελέτη και η εκμάθηση του εργαλείου Java Swing με τη γλώσσα προγραμματισμού Java. Στη συνέχεια έγινε η υλοποίηση κάποιων μικρών παραδειγμάτων για την καλύτερη κατανόηση και για να είναι σίγουρο ότι το Java Swing μπορεί να ανταπεξέλθει στις απαιτήσεις μας. Στη συνέχεια θεωρήθηκε απαραίτητο να γίνει η μελέτη και η εκμάθηση της κλάσης Graphics της Java σε συνδυασμό με το Java Swing για να προστεθεί κίνηση στη γραφική αναπαράσταση των αλγορίθμων για ένα καλύτερο αποτέλεσμα.

Τέλος, έγινε η μελέτη κάποιων επιλεγμένων παράλληλων και εύρωστων αλγορίθμων και στην πορεία έγινε η υλοποίηση της γραφικής αναπαράστασής τους. Συγκεκριμένα, αναπαραστάθηκαν ο μη βέλτιστος και ο βέλτιστος αλγόριθμος παράλληλης άθροισης [5], ο μη βέλτιστος και ο βέλτιστος εύρωστος αλγόριθμος W [6] και ο αλγόριθμος X [6].

1.3 Δομή Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία αποτελείται από εφτά κεφάλαια. Στο Κεφάλαιο 2, που ακολουθεί, γίνεται μια εισαγωγή στον παραλληλισμό και στη σημαντικότητά του, ενώ στη συνέχεια γίνεται μια περιγραφή των μοντέλων PRAM [5], F-PRAM [6,2] και A-PRAM [7,2], στα οποία σχεδιάστηκαν οι αλγόριθμοι που αναφέρθηκαν στο Υποκεφάλαιο 1.2. Στο Κεφάλαιο 3, γίνεται αναφορά στη γλώσσα προγραμματισμού Java και στα εργαλεία που χρησιμοποιήθηκαν για την εκπόνηση αυτής της εργασίας (Java Swing, Graphics, NetBeans IDE). Στα επόμενα κεφάλαια, δηλαδή στα κεφάλαια 4 μέχρι 6, περιγράφονται τα βασικά στοιχεία των αλγορίθμων που υλοποιήθηκαν και γίνεται επεξήγηση του τρόπου υλοποίησης της γραφικής τους αναπαράστασης. Πιο συγκεκριμένα, το Κεφάλαιο 4 ασχολείται με τον μη βέλτιστο και το βέλτιστο αλγόριθμο παράλληλης άθροισης στο μοντέλο PRAM, το Κεφάλαιο 5 με τον αλγόριθμο W στο μοντέλο F-PRAM και την επίλυση του προβλήματος Write-All. Με το ίδιο πρόβλημα ασχολείται και το Κεφάλαιο 6 μόνο που σε αυτό το κεφάλαιο μελετάται η επίλυση του προβλήματος αυτού με τον αλγόριθμο X στο μοντέλο A-PRAM. Η εργασία κλείνει με το Κεφάλαιο 7, στο οποίο αναφέρονται τα συμπεράσματα που εξάγονται από τη δουλειά που έγινε στα πλαίσια αυτής της διπλωματικής εργασίας.

Κεφάλαιο 2

Υπόβαθρο Μελέτης

2.1 Παράλληλος Υπολογισμός και η Σημαντικότητά του

Για αρκετά χρόνια η επιστήμη της πληροφορικής μελετούσε τη λειτουργία του σειριακού υπολογισμού. Πιο συγκεκριμένα, επικεντρωνόταν στην επίλυση προβλημάτων με τη χρήση σειριακών αλγορίθμων. Ένας επεξεργαστής είναι υπεύθυνος για τη σειριακή εκτέλεση όλων των εντολών ενός προγράμματος.

Στην πορεία όμως άρχισε να αναπτύσσεται σημαντικά και ο παράλληλος υπολογισμός. Ο παραλληλισμός ασχολείται με τη συνεταιριστική και ταυτόχρονη επεξεργασία δεδομένων από περισσότερους από ένα επεξεργαστές, αποσκοπώντας στη γρήγορη επίλυση σύνθετων υπολογιστικών προβλημάτων. Οι επεξεργαστές, για να είναι ικανοί να εκτελούν παράλληλους υπολογισμούς, πρέπει συνήθως να είναι του ίδιου τύπου για να γίνεται ισοζυγισμένη κατανομή εργασίας, να βρίσκονται σε κοντινή απόσταση για να μην υπάρχει καθυστέρηση επικοινωνίας και φυσικά να επιλύουν μαζί ένα κοινό πρόβλημα-στόχο. Μια συλλογή από επεξεργαστές ικανή να εκτελεί παράλληλους υπολογισμούς έχει τεράστια υπολογιστική δύναμη.

Η σημαντικότητα του παράλληλου υπολογισμού βασίζεται στα πλεονεκτήματα που προσφέρει σε σχέση με το σειριακό. Με τον παράλληλο υπολογισμό επιτυγχάνεται η επίλυση δύσκολων και μεγάλων υπολογιστικών προβλημάτων σε χρόνο μικρότερο από αυτόν που

χρειάζεται ο σειριακός για ένα κοινό πρόβλημα. Επίσης, μπορεί να διεκπεραιώσει πολύπλοκα προβλήματα τα οποία ο σειριακός υπολογισμός δεν μπορεί να επιλύσει. Επιπλέον, η χρήση του παράλληλου υπολογισμού είναι οικονομική και συμφέρουσα, αφού τα παράλληλα συστήματα έχουν πλέον χαμηλό κόστος [2,5,7].

Για να έχουμε όμως τις επιθυμητές επιδόσεις-ταχύτητες στον παράλληλο υπολογισμό δεν αρκεί μόνο η αλλαγή στην αρχιτεκτονική των υπολογιστών με την πρόσθεση επεξεργαστών, αλλά εισάγονται και καινούριες απαιτήσεις από το λογισμικό, για να μπορεί να αξιοποιηθεί ορθά αυτή η υπολογιστική δύναμη. Έτσι, στα παράλληλα συστήματα χρησιμοποιούνται αλγόριθμοι που είναι σχεδιασμένοι ειδικά γι' αυτά τα συστήματα και ονομάζονται *Παράλληλοι Αλγόριθμοι*. Για να μπορέσουμε να αναλύσουμε θεωρητικά και να καθορίσουμε το πόσο αποδοτικός είναι ο κάθε παράλληλος αλγόριθμος πρέπει να υπολογίσουμε την πολυπλοκότητά του [2,5,7].

Για την ανάλυση της πολυπλοκότητας ενός παράλληλου αλγορίθμου χρησιμοποιούμε τις παρακάτω έννοιες:

- Πλήθος Επεξεργαστών-P(n)

  Είναι ο αριθμός των επεξεργαστών που χρησιμοποιήθηκαν για την επίλυση ενός προβλήματος μεγέθους n.

- Παράλληλος Χρόνος Εκτέλεσης-$T_p(n)$

  Είναι ο χρόνος που χρειάζονται p επεξεργαστές, δουλεύοντας παράλληλα, να επιλύσουν ένα πρόβλημα μεγέθους n.

- Κόστος-$C_p(n)$

  Είναι το γινόμενο του παράλληλου χρόνου εκτέλεσης επί τον αριθμό των επεξεργαστών που χρησιμοποιήθηκαν, για να επιλυθεί παράλληλα ένα πρόβλημα μεγέθους n.

  $C_p(n) = P(n) * T_p(n)$

- Χρόνος Καλύτερου Σειριακού Αλγορίθμου-$T^*(n)$

  Είναι ο χρόνος που χρειάζεται ένας επεξεργαστής για να επιλύσει ένα πρόβλημα μεγέθους n, εκτελώντας τον πιο αποδοτικό σειριακό αλγόριθμο.

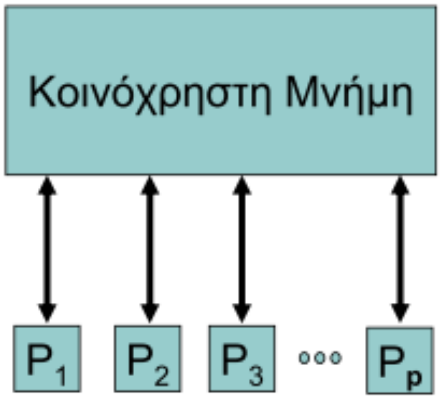
Αφού στόχος του παράλληλου υπολογισμού είναι η γρήγορη επίλυση ενός δεδομένου προβλήματος με το μικρότερο δυνατό χρόνο, αυτό σημαίνει ότι επιθυμούμε το *Βέλτιστο Παράλληλο Αλγόριθμο*. Για να είναι κάποιος παράλληλος αλγόριθμος βέλτιστος, πρέπει το κόστος εκτέλεσής του να είναι της τάξης του χρόνου εκτέλεσης του πιο γρήγορου σειριακού

αλγορίθμου που επιλύει το ίδιο πρόβλημα. Συγκεκριμένα, ένας αλγόριθμος είναι βέλτιστος όταν $C_p(n)=O(T^*(n))$.

2.2 Μοντέλο PRAM

Η χρήση πολλαπλών επεξεργαστών πάνω στο ίδιο υπολογιστικό σύστημα που δουλεύουν παράλληλα για το ίδιο πρόβλημα δημιουργεί επιπρόσθετες απαιτήσεις στη μοντελοποίηση και στο σχεδιασμό των αλγορίθμων. Για παράδειγμα, πρέπει να ληφθεί υπόψη πως γίνεται η ανάθεση εργασιών στους επεξεργαστές, που αποθηκεύονται τα δεδομένα εισόδου, τα ενδιάμεσα αποτελέσματα και τα δεδομένα εξόδου, πως οι επεξεργαστές έχουν πρόσβαση σε αυτά και πως επιτυγχάνεται η επικοινωνία μεταξύ τους.

Στην εργασία αυτή θα επικεντρωθούμε στην αρχιτεκτονική που βασίζεται στην κοινόχρηστη μνήμη και πιο συγκεκριμένα στο μοντέλο PRAM [5]. Το PRAM (**P**arallel **R**andom **A**ccess **M**achine) είναι ένα μοντέλο κοινόχρηστης μνήμης που αποτελείται από ένα μέγιστο αριθμό επεξεργαστών, ο καθένας από τους οποίους έχει και τη δική του τοπική μνήμη (με χρόνο πρόσβασης Θ(1)). Κατατάσσεται στην κατηγορία των μοντέλων παράλληλου υπολογισμού SIMD (Single Instruction stream, Multiple Data stream), όπου σε κάθε βήμα όλοι οι επεξεργαστές εκτελούν την ίδια εντολή αλλά με διαφορετικά δεδομένα.



Σχήμα 2.1 – Μοντέλο PRAM [2]

Η κοινόχρηστη μνήμη, στο μοντέλο PRAM, χρησιμεύει για την έμμεση επικοινωνία μεταξύ των επεξεργαστών, αφού εδώ αποθηκεύονται μεταβλητές οι οποίες είναι ορατές από όλους τους διαθέσιμους επεξεργαστές. Συγκεκριμένα, στην κοινόχρηστη μνήμη αποθηκεύονται όσα αναφέρθηκαν στην αρχή, δηλαδή, τα δεδομένα εισόδου του προβλήματος το οποίο καλούμαστε να λύσουμε, τα ενδιάμεσα αποτελέσματα που επιθυμούν να είναι ορατά στους

υπόλοιπους επεξεργαστές και τα αποτελέσματα εξόδου. Ο κάθε επεξεργαστής μπορεί να γράψει ένα μήνυμα σε μια κυψελίδα της κοινόχρηστης μνήμης και οποιοσδήποτε άλλος μπορεί να το διαβάσει και να το επεξεργαστεί. Οι επεξεργαστές έχουν ομοιόμορφη πρόσβαση στην κοινόχρηστη μνήμη σε σταθερό χρόνο Θ(1). Επίσης, όταν εκτελείται μια εντολή από κάποιο επεξεργαστή, τότε η ίδια εντολή εκτελείται από όλους τους υπόλοιπους αλγορίθμους την ίδια χρονική στιγμή. Δηλαδή, η επεξεργασία είναι *συγχρονισμένη*. Είναι όμως δυνατό κάποιοι επεξεργαστές να μην εκτελούν κάποια εντολή για ένα χρονικό διάστημα, επειδή αυτή δε χρειάζεται για τη σωστή επίλυση του προβλήματος. Σ' αυτή την περίπτωση θεωρούμε ότι οι επεξεργαστές εκτελούν την εντολή no-op (no operation) [2, 5].

Υπάρχουν διάφοροι τύποι PRAM που αφορούν τους περιορισμούς ταυτόχρονης πρόσβασης στην κοινόχρηστη μνήμη. Οι τύποι αυτοί είναι οι ακόλουθοι [2, 5]:

- EREW: Exclusive Read, Exclusive Write

  Δεν επιτρέπεται η ταυτόχρονη ανάγνωση ή γραφή της ίδιας κυψελίδας της κοινόχρηστης μνήμης από περισσότερους από ένα επεξεργαστές. Είναι το πιο περιοριστικό (αδύνατο) μοντέλο.
- CREW: Concurrent Read, Exclusive Write

  Μπορούν όλοι οι επεξεργαστές να διαβάσουν από την ίδια κυψελίδα μνήμης ταυτόχρονα, αλλά μόνο ένας επεξεργαστής μπορεί να γράψει στην ίδια κυψελίδα μνήμης.
- ERCW: Exclusive Read, Concurrent Write

  Μπορούν όλοι οι επεξεργαστές να γράψουν στην ίδια κυψελίδα μνήμης ταυτόχρονα, αλλά μόνο ένας επεξεργαστής μπορεί να διαβάσει από την ίδια κυψελίδα μνήμης. Το μοντέλο αυτό υπάρχει μόνο σε θεωρητικό επίπεδο.
- CRCW: Concurrent Read, Concurrent Write

  Δεν υπάρχουν καθόλου περιορισμοί που να αφορούν την ταυτόχρονη πρόσβαση των επεξεργαστών στην ίδια κυψελίδα της κοινόχρηστης μνήμης. Είναι το πιο ελαστικό (δυνατό) μοντέλο.

  Στην περίπτωση όπου υπάρχει ταυτόχρονη ανάγνωση της ίδιας κυψελίδας μνήμης, από περισσότερους από ένα επεξεργαστές, τότε δεν παρουσιάζεται πρόβλημα, αφού ο κάθε επεξεργαστής απλά "βλέπει" την τιμή της κοινόχρηστης κυψελίδας και την αποθηκεύει στην τοπική του μνήμη. Όταν, όμως, στην ίδια κυψελίδα μνήμης υπάρχει ταυτόχρονη γραφή από διαφορετικούς επεξεργαστές, τότε υπάρχει η πιθανότητα να

δημιουργηθούν προβλήματα. Έτσι, το CRCW PRAM έχει χωριστεί σε υπομοντέλα για να χειριστούν την ταυτόχρονη γραφή, χωρίς να υπάρχουν προβλήματα. Τα μοντέλα αυτά είναι τα ακόλουθα:

- Common CRCW PRAM

  Όλοι οι επεξεργαστές που γράφουν την ίδια στιγμή στην ίδια κυψελίδα της κοινόχρηστης μνήμης πρέπει να γράψουν την ίδια τιμή.

- Arbitrary CRCW PRAM

  Ένας από τους επεξεργαστές που επιθυμούν να κάνουν εγγραφή στην ίδια κυψελίδα κοινόχρηστης μνήμης επιλέγεται τυχαία και αποθηκεύει τη δική του τιμή, ενώ οι τιμές των άλλων επεξεργαστών ακυρώνονται.

- Priority CRCW PRAM

  Επιλέγεται από τους επεξεργαστές που επιθυμούν να κάνουν εγγραφή στην ίδια κυψελίδα κοινόχρηστης μνήμης ο επεξεργαστής με τη μεγαλύτερη προτεραιότητα για να γράψει την δική του τιμή, ενώ οι τιμές των άλλων επεξεργαστών ακυρώνονται. Η προτεραιότητα συνήθως καθορίζεται από το αναγνωριστικό των επεξεργαστών, δηλαδή το μοναδικό προσωπικό τους αριθμό.

Τα διάφορα μοντέλα του PRAM που περιγράφηκαν προηγουμένως έχουν διαφορετική δυναμικότητα και περιορισμούς. Η ιεραρχία δυναμικότητας των μοντέλων του PRAM παρατίθεται πιο κάτω ξεκινώντας από το πιο περιοριστικό (αδύναμο) μοντέλο και καταλήγοντας στο πιο ελαστικό (δυνατό):

EREW→ CREW→ Common CRCW→ Arbitrary CRCW→ Priority CRCW

Όταν ένας αλγόριθμος σχεδιαστεί σε ένα πιο περιοριστικό μοντέλο, τότε μπορεί να υλοποιηθεί με την ίδια πολυπλοκότητα χρόνου και κόστους σε οποιοδήποτε πιο ελαστικό μοντέλο. Όταν, όμως, ένας αλγόριθμος είναι σχεδιασμένος σε πιο ελαστικό μοντέλο, τότε δύναται να υλοποιηθεί σε πιο περιοριστικό μοντέλο μόνο με κάποιο μεγαλύτερο κόστος ή με μεγαλύτερο χρόνο εκτέλεσης.

Για το λόγο αυτό είναι απαραίτητο όταν κάποιος σχεδιάζει ένα αλγόριθμο στο μοντέλο PRAM να έχει υπόψη του τον τύπο του μοντέλου στο οποίο θα υλοποιηθεί ο αλγόριθμος, καθώς και τους περιορισμούς του μοντέλου αυτού.

2.3 Μοντέλο F-PRAM

Το μοντέλο F-PRAM [2,6] κληρονομεί πολλά χαρακτηριστικά του από το μοντέλο PRAM που περιγράφηκε στο προηγούμενο υποκεφάλαιο. Σύμφωνα, λοιπόν, με αυτά που αναφέρθηκαν πιο πριν, στο μοντέλο PRAM υπάρχουν πανομοιότυποι επεξεργαστές που εργάζονται όλοι παράλληλα και συγχρονισμένα για την ολοκλήρωση ενός προβλήματος, χωρίς οι επεξεργαστές να υπόκεινται σε σφάλματα. Αυτή λοιπόν, είναι και η μόνη σημαντική διαφορά του μοντέλου PRAM με το μοντέλο F-PRAM, αφού το μοντέλο F-PRAM είναι σχεδιασμένο να ανταπεξέλθει σε πιθανή κατάρρευση κάποιου επεξεργαστή κατά τη διάρκεια εκτέλεσης του παράλληλου υπολογισμού.

Τα σφάλματα κατάρρευσης συμβαίνουν όταν ένας επεξεργαστής μπορεί να σταματήσει να λειτουργεί σε οποιοδήποτε σημείο του υπολογισμού και, όταν σταματήσει, δεν επιστρέφει ξανά στον υπολογισμό. Δεδομένου κάποιου υπολογισμού, όπως για παράδειγμα επίλυσης ενός προβλήματος, λέμε ότι ο υπολογισμός έγκειται σε f σφάλματα κατάρρευσης, αν από την αρχή του υπολογισμού μέχρι το τέλος του, μέχρι την επίλυση του προβλήματος, καταρρεύσουν το πολύ f διαφορετικοί επεξεργαστές. Απαραίτητη όμως προϋπόθεση είναι μέχρι το τέλος να υπάρχει έστω ένας επεξεργαστής, οποιοσδήποτε, που να μην έχει καταρρεύσει, για να μπορέσει να ολοκληρώσει την επίλυση του προβλήματος και να φθάσει στο τέλος του υπολογισμού.

Ένας παράλληλος αλγόριθμος σχεδιασμένος για ένα συγκεκριμένο πρόβλημα που επιλύει το πρόβλημα αποδοτικά και ταυτόχρονα ανέχεται σφάλματα επεξεργαστών ονομάζεται *εύρωστος* [2,6].

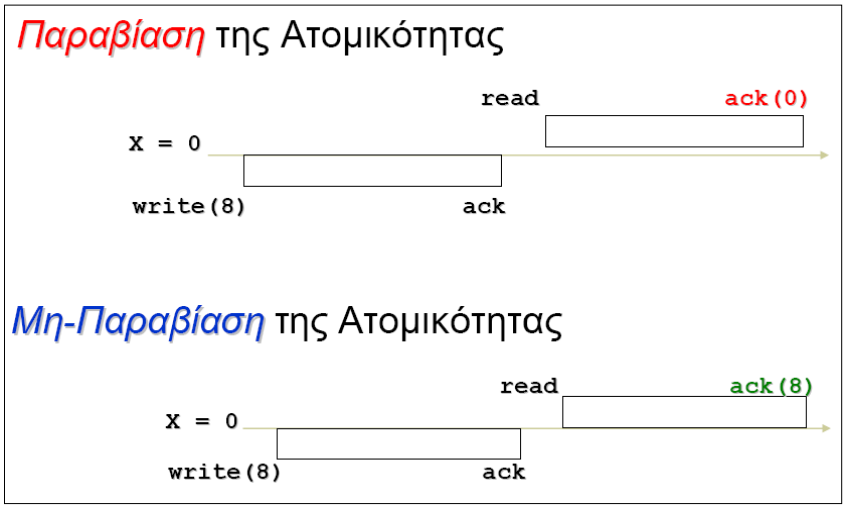
2.4 Μοντέλο A-PRAM

Το μοντέλο A-PRAM [2,6] είναι υπομοντέλο του μοντέλου παράλληλου υπολογισμού MIMD (Multiple Instruction stream, Multiple Data stream) και όχι του SIMD. Στο μοντέλο A-PRAM οι επεξεργαστές ενεργούν εντελώς ασυγχρόνιστα, δηλαδή δεν εκτελούν σε κάθε βήμα όλοι οι επεξεργαστές την ίδια εντολή. Ενώ στα μοντέλα PRAM και F-PRAM που υπάγονται στο μοντέλο SIMD, όπως προαναφέρθηκε, εργάζονται συγχρονισμένα, που σημαίνει ότι σε κάθε βήμα όλοι οι επεξεργαστές εκτελούν την ίδια εντολή. Επίσης, στο A-PRAM οι επεξεργαστές μπορούν να γράψουν και να διαβάσουν σε οποιαδήποτε κυψελίδα

της κοινόχρηστης μνήμης, όμως κάθε φορά μπορεί να πάρει διαφορετικό χρόνο η πρόσβαση στη μνήμη. Βασικά, σε αυτό το μοντέλο ο χρόνος δεν παίζει ουσιαστικό ρόλο στο σχεδιασμό αλγορίθμων, ενώ είναι δύσκολο να διακρίνει κανείς αν κάποιος επεξεργαστής έχει καταρρεύσει ή είναι πολύ αργός [2,6].

Επιπρόσθετα, στο μοντέλο αυτό δεν υπάρχει ταυτόχρονη πρόσβαση σε μια κυψελίδα της κοινόχρηστης μνήμης, αλλά υπάρχει η λεγόμενη *ατομική πρόσβαση*, όπου κάθε ακολουθία ασυγχρόνιστων προσβάσεων από τον ίδιο ή και διαφορετικούς επεξεργαστές σε μια συγκεκριμένη κυψελίδα μνήμης μπορεί να τοποθετηθεί σε μερική διάταξη. Με αυτό τον τρόπο η τιμή που θα επιστρέφεται κατά την ανάγνωση από μια κυψελίδα της κοινόχρηστης μνήμης θα είναι η τιμή της τελευταίας γραφής, η οποία έχει ολοκληρωθεί ή η τιμή μιας γραφής η οποία συμβαίνει την ίδια χρονική στιγμή με την ανάγνωση και δεν έχει ολοκληρωθεί ακόμη. Παρομοίως, η τιμή που θα επιστραφεί κατά την επόμενη ανάγνωση της κυψελίδας είτε θα είναι ίδια με την προηγούμενη ανάγνωση είτε θα έχει την τιμή μιας γραφής που εκτελέστηκε μετά από την προηγούμενη ανάγνωση. Άρα, αν δεν είμαστε προσεκτικοί, η ατομικότητα μπορεί να επιφέρει απροσδιοριστία αποτελεσμάτων [2,6].

Στην συνέχεια θα ακολουθήσει ένα παραδείγματα, το οποίο μας δείχνει την περίπτωση όπου υπάρχει παραβίαση της ατομικότητας, με αποτέλεσμα να έχουμε απροσδιοριστία των αποτελεσμάτων και την περίπτωση όπου δεν υπάρχει παραβίαση της ατομικότητας. Σύμφωνα με το παράδειγμα αυτό την πρώτη φορά υπάρχει παραβίαση της ατομικότητας γιατί στο τέλος ο επεξεργαστής διαβάζει την τιμή 0 ενώ στην δεύτερη περίπτωση δεν υπάρχει παραβίαση αφού διαβάζει κανονικά την τιμή 8.



Σχήμα 2.2 – Παράδειγμα Ατομικότητας A-PRAM [2]

Κεφάλαιο 3

Γλώσσα Προγραμματισμού Java και Βοηθητικά Εργαλεία

3.1 Γλώσσα Προγραμματισμού Java

Η Java είναι μία γλώσσα προγραμματισμού και υπολογιστική πλατφόρμα που κυκλοφόρησε από την Sun Microsystems το 1995 [8]. Είναι η τεχνολογία που ενισχύει προγράμματα state-of-the-art, όπως παιχνίδια και επιχειρηματικές εφαρμογές. Υπολογίζεται πως η Java τρέχει σε περισσότερους από 850 εκατομμύρια προσωπικούς ηλεκτρονικούς υπολογιστές και σε δισεκατομμύρια συσκευές σε όλο τον κόσμο [8].

Υπάρχουν πολλές εφαρμογές και ιστοσελίδες που δεν θα λειτουργούσαν αν δεν ήταν εγκαταστημένη στον υπολογιστή η Java. Επιπλέον, είναι μία γλώσσα υψηλού επιπέδου που μας προσφέρει απλότητα, αντικειμενοστρέφεια, ανεξαρτησία αρχιτεκτονικής και φορητότητα, δυναμικότητα, πολυνηματικότητα, κατανεμημένη, υψηλή επίδοση και ασφάλεια.

Η προγραμματιστική διαπροσωπεία εφαρμογών της Java (Java Application Programming Interface - API) αποτελείται από μία μεγάλη συλλογή έτοιμων δομοστοιχείων λογισμικού, τα οποία υλοποιούν διάφορες χρήσιμες λειτουργικότητες. Το API είναι οργανωμένο σε λογικές

συλλογές (βιβλιοθήκες) κλάσεων και διαπροσωπείων, οι οποίες στην ορολογία της Java, αποκαλούνται πακέτα (packages) [9].

Σ' αυτή την εργασία, με την επιλογή της Java, έχουμε τη δυνατότητα ανάπτυξης γραφικής διαπροσωπείας (GUI) με το εργαλείο Java Swing [11] και δημιουργίας γραφικής αναπαράστασης με κίνηση με το Java Graphics API [1,12]. Επίσης, κάποια άλλα πλεονεκτήματα που μας παρέχει η Java που βοήθησαν στην εργασία αυτή, ήταν η παροχή έτοιμων βιβλιοθηκών, η εύκολη διαχείριση της μνήμης και η ανεξαρτησία πλατφόρμας.

3.2 Εργαλείο NetBeans IDE

Το NetBeans Integrated Development Environment (IDE) [10] είναι μία ολοκληρωμένη εφαρμογή ανοιχτού κώδικα για παραγωγή προγραμμάτων, διαθέσιμη για πολλά λειτουργικά συστήματα, όπως Windows, Mac, Linux και Solaris. Επίσης το NetBeans αποτελείται από μια πλατφόρμα εφαρμογών που επιτρέπει στους προγραμματιστές να δημιουργήσουν εφαρμογές που αφορούν ιστοσελίδες, προγράμματα επιχειρήσεων, desktop εφαρμογές και εφαρμογές για κινητά τηλέφωνα, χρησιμοποιώντας τη πλατφόρμα Java ή ακόμα γλώσσες προγραμματισμού όπως PHP, JavaScript, Groovy, C/C++, κ.ά. προσφέροντάς τους όλα τα εργαλεία που χρειάζονται [10].

Στην περίπτωσή μας, το NetBeans επιλέχθηκε για την ικανότητά του να προγραμματίζει σε Java και να κάνει πιο απλή την ανάπτυξη εφαρμογών Java Swing, χρησιμοποιώντας τα εργαλεία που προσφέρει στον προγραμματιστή για να δημιουργήσει γραφική διαπροσωπεία.

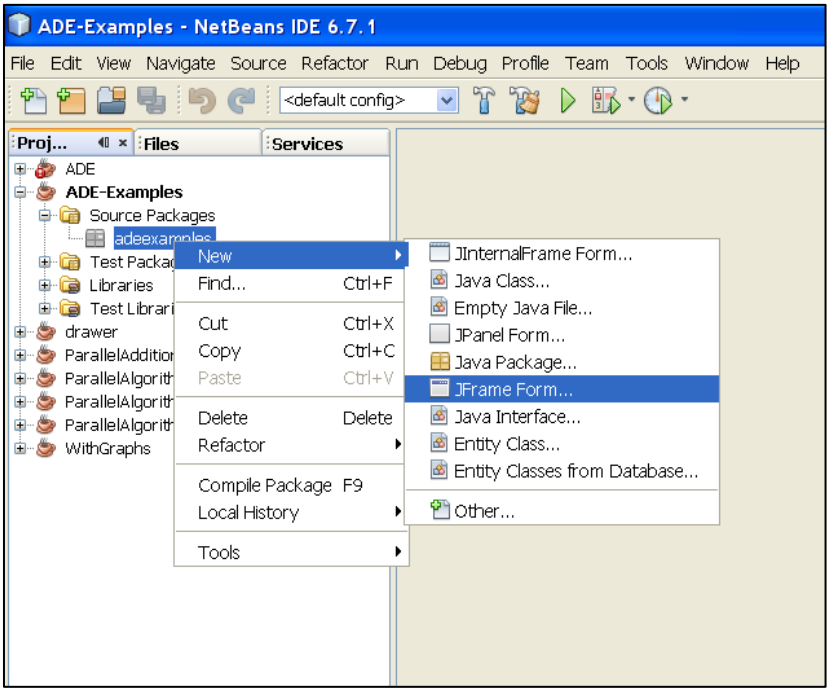
3.3 Εργαλείο Java Swing

Οι κλάσεις Swing (μέρος από το λογισμικό Java™ Foundation Classes (JFC)) [11] παρέχουν ένα σύνολο από συστατικά για την ανάπτυξη γραφικών διεπαφών (GUIs) και προσθέτουν πλούσια λειτουργικά γραφικά και διαδραστικότητα με εφαρμογές της Java. Τα Swing components είναι εξολοκλήρου γραμμένα στη γλώσσα προγραμματισμού Java. Επίσης το Java Swing επιτρέπει τη δημιουργία GUIs που μπορούν είτε να μοιάζουν το ίδιο σε όλες τις πλατφόρμες είτε να προσομοιώνουν την εμφάνιση και την αίσθηση της πλατφόρμας του λειτουργικού συστήματος στο οποίο τρέχουν (όπως τα Microsoft Windows, SolarisTM ή Linux) [11].

3.4 Παραδείγματα Java Swing

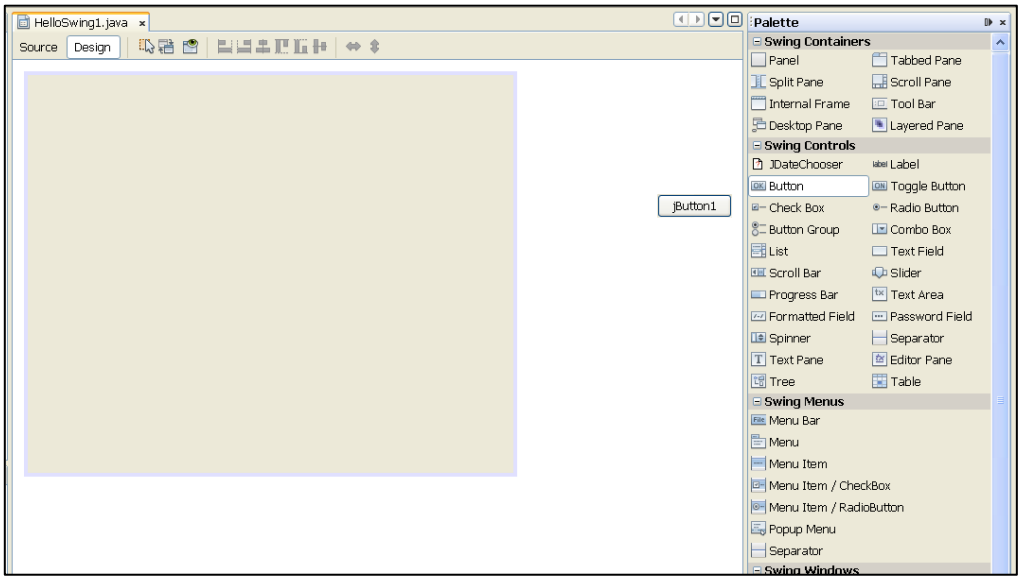
Εάν κάποιος επιθυμεί να δημιουργήσει μία γραφική διαπροσωπεία στο Java Swing έχει την επιλογή να χρησιμοποιήσει κάποιο εργαλείο όπως το NetBeans IDE στο οποίο μπορεί να επιλέξει τη δημιουργία μιας φόρμας (π.χ. JFrame) και στη συνέχεια να τοποθετήσει διάφορα συστατικά πάνω σε αυτή, «σέρνοντάς» τα από την παλέτα και τοποθετώντας τα στη θέση που επιθυμεί πάνω στη φόρμα. Κατόπιν, το NetBeans από μόνο του θα γράψει τον απαραίτητο κώδικα.

Μπορεί όμως κάποιος να επιθυμεί να γράψει από μόνος του τον κώδικα υλοποίησης ή να το απαιτεί η εφαρμογή που θέλει να υλοποιήσει, όπως για παράδειγμα αν η εφαρμογή του θέλει να τοποθετεί δυναμικά κάποια components στη φόρμα, τα οποία θα ζητά ο χρήστης. Ακολουθούν μερικά μικρά παραδείγματα και για τις δύο περιπτώσεις [3,4].

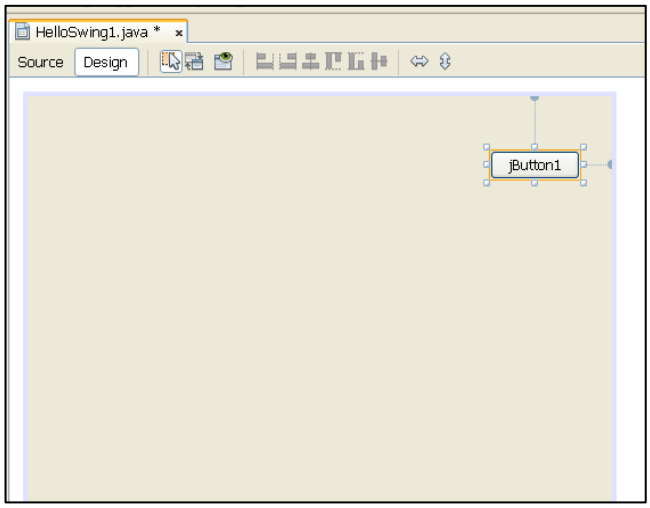


Σχήμα 3.1 – Δημιουργία φόρμας JFrame με την χρήση του εργαλείου NetBeans IDE

Στο πιο πάνω σχήμα (Σχήμα 3.1) βλέπουμε ένα παράδειγμα για το πόσο απλά μπορούμε να δημιουργήσουμε μία φόρμα με τη χρήση του NetBeans IDE. Στα επόμενα σχήματα (Σχήμα 3.2 και Σχήμα 3.3) θα δούμε την φόρμα που δημιούργησε το NetBeans, καθώς και το πώς βάζουμε ένα κουμπί (OK Button) στη φόρμα σέρνοντας το από την παλέτα.



Σχήμα 3.2 – Εμφάνιση νέας φόρμας JFrame



Σχήμα 3.3 – Εμφάνιση φόρμας με κουμπί (OK Button)

```
private void initComponents() {

    jButton1 = new javax.swing.JButton();

    setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

    jButton1.setText("jButton1");

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
            .addContainerGap(405, Short.MAX_VALUE)
            .addComponent(jButton1)
            .addGap(29, 29, 29))
    );
    layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addGap(49, 49, 49)
            .addComponent(jButton1)
            .addContainerGap(347, Short.MAX_VALUE))
    );

    pack();
}// </editor-fold>
```

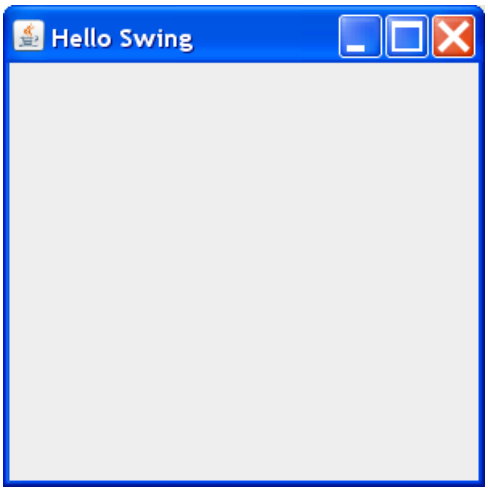
Σχήμα 3.4 – Κώδικας που δημιούργησε το NetBeans IDE

Στο πιο πάνω σχήμα (Σχήμα 3.4), βλέπουμε ένα κομμάτι από τον κώδικα που δημιούργησε το NetBeans για τη δημιουργία της φόρμας. Με κόκκινο είναι επιλεγμένες οι δύο γραμμές του κώδικα που πρόσθεσε για τη δημιουργία του κουμπιού. Με τον ίδιο τρόπο μπορούμε να προσθέσουμε και άλλα components στη φόρμα, όπως labels, panels, combo box κ.ά.

Όπως αναφέρθηκε προηγουμένως, μπορούμε να δημιουργήσουμε φόρμα γράφοντας από μόνοι μας το κώδικα. Στον κώδικα που ακολουθεί γίνεται η δημιουργία μίας φόρμας JFrame, η οποία έχει τίτλο Hello Swing.

```
import javax.swing.*;
public class HelloSwing {
    public static void main (String[] args){
        JFrame frame = new JFrame ("Hello Swing");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}
```

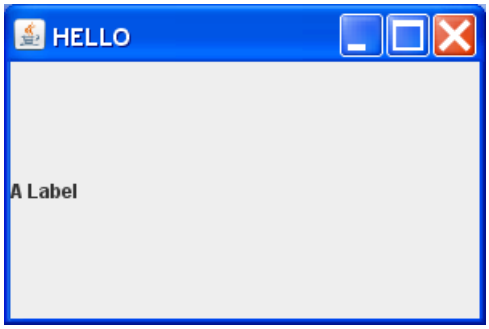
Όταν εκτελέσουμε το πιο πάνω κομμάτι κώδικα τότε θα μας εμφανιστεί η φόρμα που φαίνεται στο επόμενο σχήμα 3.5.



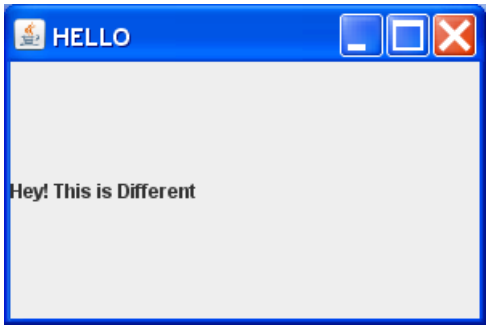
Σχήμα 3.5 – Φόρμα JFrame, υλοποίηση με κώδικα

Στη συνέχεια, θα δούμε ένα κομμάτι κώδικα στο οποίο δημιουργούμε μια φόρμα JFrame με τίτλο HELLO, διαστάσεων 300×200, στην οποία τοποθετούμε μια ετικέτα (label) με αρχικό κείμενο «A Label», το οποίο καθώς τρέχει το πρόγραμμα αλλάζει και γίνεται «Hey! This is different».

```
import javax.swing.*;
import java.util.concurrent.*;

public class HelloLabel{
    public static void main (String[] args) throws Exception {
    JFrame frame =new JFrame ("HELLO");
    JLabel label = new JLabel ("A Label");
    frame.add(label);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(300, 200);
    frame.setVisible(true);
    TimeUnit.SECONDS.sleep(1);
    label.setText("Hey! This is Different");
    }
}
```



Σχήμα 3.6 – Εκτέλεση κώδικα HelloLabel.java (μέρος1ο)



Σχήμα 3.7 – Εκτέλεση κώδικα HelloLabel.java (μέρος 2ο)

Στα δύο πιο πάνω σχήματα (Σχήμα 3.6 και Σχήμα 3.7) βλέπουμε το αποτέλεσμα της εκτέλεσης του πιο πάνω κώδικα (HelloLabel.java). Αρχικά διαβάζουμε το μήνυμα «A Label» και στη συνέχεια το μήνυμα «Hey! This is different».

3.5 Java Graphics API

Η διαπροσωπεία Java 2D (Java 2D™ API) [12] είναι ένα σύνολο από κλάσεις για σχεδίαση και αναπαράσταση δισδιάστατων σχεδιαγραμμάτων και εικόνων. Επιτρέπει την δημιουργία γραμμικών σκίτσων, κείμενων και αναπαράσταση εικόνων σε ένα μοναδικό περιεκτικό μοντέλο. Το API παρέχει εκτεταμένη υποστήριξη για σύνθεση απλών εικόνων και εικόνων με μεταβλητό δείκτη διαφάνειας, ένα σύνολο από κλάσεις για να ορίζεις και να μετατρέπεις χρωματικά μοντέλα, καθώς και ένα πλούσιο σύνολο από τελεστές για επεξεργασία εικόνας. Αυτές οι κλάσεις βρίσκονται στα πακέτα java.awt και java.awt.image. Δύο βασικές κλάσεις της διαπροσωπείας που χρησιμοποιήθηκαν σ' αυτή την εργασία, είναι η Graphics και Graphics2D που βρίσκονται στο java.awt.Graphics και στο java.awt.Graphics2D αντίστοιχα [12].

Η κλάση Graphics είναι η βασική κλάση για όλα τα graphics contexts που επιτρέπουν σε μια εφαρμογή να σχεδιάσει επάνω σε συστατικά. Ένα αντικείμενο τύπου Graphics κρατάει πληροφορίες που χρειάζονται για τις βασικές λειτουργίες rendering που υποστηρίζει η Java, όπως είναι το τρέχον χρώμα, φόντο, οι συντεταγμένες κ.ά. Η κλάση Graphics2D επεκτείνει την κλάση Graphics για να παρέχει περισσότερο και καλύτερο έλεγχο στη γεωμετρία, μετασχηματισμό συντεταγμένων, διαχείριση χρωμάτων και διάταξη κειμένου. Αυτή είναι η κλάση για απόδοση-σχεδίαση δισδιάστατων σχημάτων, κειμένου και εικόνων στην Java.

3.6 Παράδειγμα με Java Graphics

Σ' αυτό το υποκεφάλαιο θα δούμε ένα παράδειγμα από κώδικα για την κλάση Graphics και στη συνέχεια θα δούμε το αποτέλεσμα που μας δίνει όταν το εκτελέσουμε.

Στο παράδειγμα που ακολουθεί συνδυάσαμε Java Swing με Graphics. Έχουμε δημιουργήσει μία φόρμα (JFrame) στο NetBeans IDE, στην οποία υπάρχουν 2 κουμπιά (ok-buttons), start και stop. Με το πάτημα του «start» ξεκινάει η κίνηση στην γραφική αναπαράσταση, όπου υπάρχουν 2 τετράγωνα, ένα μπλε και ένα κόκκινο, που ξεκινούν να κινούνται στον άξονα x. Με το πάτημα του «stop» η κίνηση σταματά και αν πατήσουμε ξανά το «start» η κίνηση των τετραγώνων ξεκινά από την αρχή.

Στο κομμάτι του κώδικα που ακολουθεί είναι ο τρόπος με τον οποίο δημιουργούνται τα δύο τετράγωνα και σχεδιάζονται, ρυθμίζοντας το χρώμα τους και τις συντεταγμένες τους. Οι μεταβλητές x1,y1,x2,y2 είναι οι μεταβλητές που ρυθμίζουν τη θέση των τετραγώνων στη φόρμα (συντεταγμένες στους άξονες x και y).

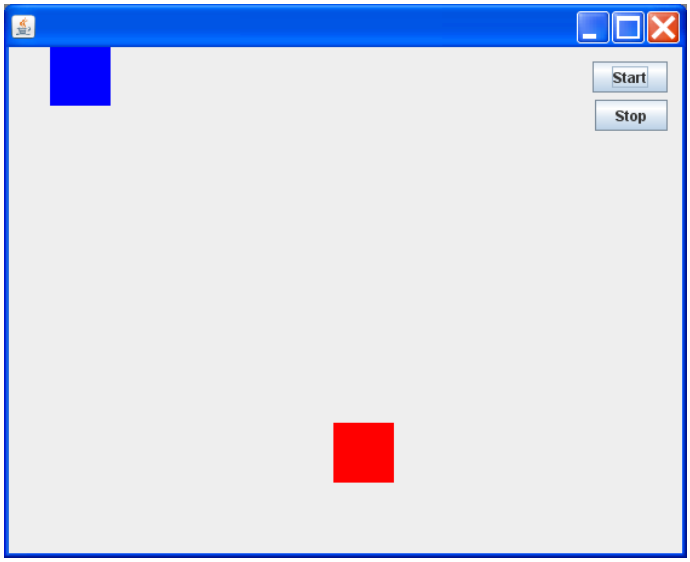
```
public void paint(Graphics g){
	//Called when the form needs to redraw itself
	Image iBuffer;
	Graphics buffer;
	//Create a buffer
	iBuffer = createImage(this.getWidth(), this.getHeight());
	buffer = iBuffer.getGraphics();
	//Draw the controls and background of the form to the buffer
	super.paintComponents(buffer);
	//Draw Blue square
	buffer.setColor(Color.blue);
	buffer.fillRect(x1, y1, 50, 50);
	//Draw Red square
	buffer.setColor(Color.red);
	buffer.fillRect(x2, y2, 50, 50);
	//Copy the buffer to the screen
	g.drawImage(iBuffer, 0, 0, this);
}
```

Στο επόμενο κομμάτι κώδικα θα δούμε πως με τη χρήση του Runnable και των threads δημιουργείται η κίνηση των τετραγώνων στον άξονα χ.

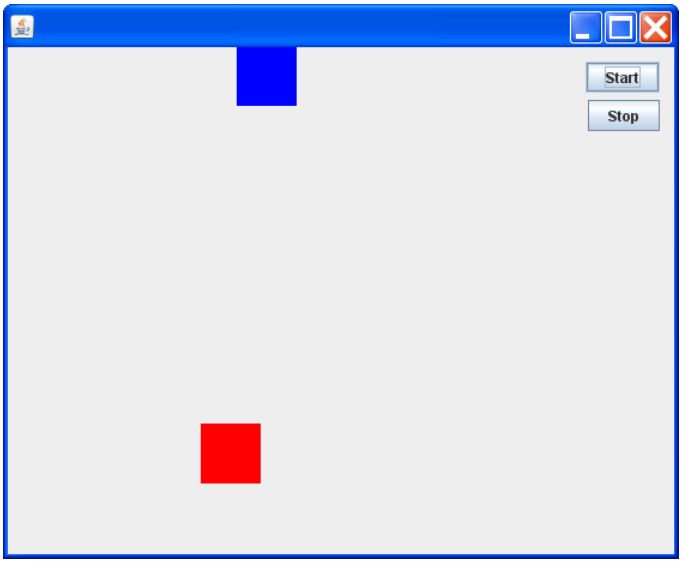
```
public class GraphicsExample extends javax.swing.JFrame implements Runnable
{
      int x1, x2, y1, y2;
      Thread runner = null;
      public void run() {
      //This function is a thread its job is to update the
      //positions of the objects and call the form's paint method

              while (runner != null) {
                  //update
                  x1 += 1;
                  x2 -= 1;
                  if (x1 > 400) {
                      x1 = -40; }
                  if (x2 < -40) {
                      x2 = 390; }
                  repaint();
                  try {
                      Thread.sleep(20);
                  } catch (InterruptedException e) {
                      // do nothing
                  }
              }
       }
      …

      private void btnStopActionPerformed(…) {
      //Kill the timer thread

              if (runner != null && runner.isAlive()) {
                  runner.stop();
              }
              runner = null;
      }

      private void btnStartActionPerformed (…){

      //When the user presses the start button reset the positions //of the
      objects and create a timer thread
              if (runner == null) {
                  x1 = 10;
                  y1 = 35;
                  x2 = 350;
                  y2 = 350;
                  runner = new Thread(this);
                  runner.start();
              }
      }
      …
}
```

Η μέθοδος btnStartActionPerformed(…) καλείται όταν ο χρήστης πατήσει το κουμπί «start» και δίνει τιμή στις μεταβλητές x1,y1,x2,y2, βάζοντας έτσι τα τετράγωνα στις αρχικές τους θέσεις. Στη συνέχεια, δημιουργεί το νήμα (thread) και καλείται η μέθοδος run(…). Η μέθοδος run(…) κάθε φορά που καλείται από το «timer thread» αλλάζει τις συντεταγμένες x1

και x2 για να αλλάζει η θέση των τετραγώνων και καλεί τη μέθοδο paint, η οποία αποτελείται από το προηγούμενο κομμάτι κώδικα που μελετήσαμε. Η μέθοδος btnStopActionPerformed(…), σταματάει την κίνηση των δύο τετραγώνων με το πάτημα του κουμπιού «stop».



Σχήμα 3.8 – Εκτέλεση κώδικα GraphicsExample.java (μέρος1^{ο})



Σχήμα 3.9 – Εκτέλεση κώδικα GraphicsExample.java (μέρος2^{ο})

Στα πιο πάνω σχήματα (Σχήμα 3.8 και Σχήμα 3.9) βλέπουμε το αποτέλεσμα που μας δίνει το παράδειγμα που σχολιάσαμε και αναλύσαμε προηγουμένως, όταν το τρέξουμε. Τα Σχήματα 3.8 και 3.9 είναι δύο στιγμιότυπα που πήραμε κατά την εκτέλεση του προγράμματος και δείχνουν την κίνηση των τετραγώνων.

Κεφάλαιο 4

Αλγόριθμος Παράλληλης Άθροισης

4.1 Πρόβλημα

Στο πρόβλημα της άθροισης, το ζητούμενο είναι ο υπολογισμός του αθροίσματος S, δεδομένης μίας λίστας n αριθμών, α_1, α_2, … α_n.

4.2 Σειριακός Αλγόριθμος

Το πρόβλημα της άθροισης λύνεται εύκολα σειριακά με τη χρήση ενός επεξεργαστή, ο οποίος διαβάζει ένα προς ένα τους αριθμούς n της λίστας και τους προσθέτει στο ολικό άθροισμα. Ακολουθεί ένα κομμάτι κώδικα, όπου γίνεται η σειριακή άθροιση των n αριθμών, στην οποία ο επεξεργαστής εκτελεί n-1 διαδοχικές προσθέσεις.

```
set S = α1
for j = 2 to n do
      S = S + αj
end do
```

Ο χρόνος εκτέλεσης του σειριακού αλγορίθμου είναι Θ( ). Όπως φαίνεται στο πιο πάνω κομμάτι κώδικα, αρχικά διαβάζεται το πρώτο στοιχείο της λίστας και αποθηκεύεται στο άθροισμα *S*, για να προστεθούν στη συνέχεια σε αυτό τα υπόλοιπα στοιχεία. Το βήμα αυτό απαιτεί σταθερό χρόνο Θ(1). Στη συνέχεια, με τη χρήση βρόγχου διαβάζεται κάθε φορά ένας αριθμός από τη λίστα και προστίθεται στο άθροισμα S, σε σταθερό χρόνο Θ(1). Το περιεχόμενο του βρόγχου εκτελείται n-1 φορές έτσι ώστε να διαβαστούν τα n-1 στοιχεία που δεν είχαν διαβαστεί στο πρώτο βήμα. Έτσι, η πολυπλοκότητα εκτέλεσης του σειριακού αλγορίθμου άθροισης είναι Θ( ).

4.3 Παράλληλος Αλγόριθμος

4.3.1 Περιγραφή Αλγορίθμου

Το πρόβλημα της παράλληλης άθροισης λύνεται και με παράλληλο αλγόριθμο, χρησιμοποιώντας περισσότερους από ένα επεξεργαστές. Συγκεκριμένα, για τον υπολογισμό αθροίσματος μίας λίστας n αριθμών χρησιμοποιούνται n/2 επεξεργαστές. Τα στοιχεία της λίστας βρίσκονται αποθηκευμένα σε n συνεχόμενες κυψελίδες της κοινόχρηστης μνήμης. Έτσι, η λίστα τυγχάνει επεξεργασίας ως πίνακας με στοιχεία στις θέσεις 1 μέχρι n [2,5].

Στη πρώτη εκτέλεση του βρόγχου του αλγορίθμου ο κάθε επεξεργαστής P_i όπου $0 < i \leq n/2$ διαβάζει και αθροίζει δύο διαφορετικούς αριθμούς από τις θέσεις $\alpha_{(2i-1)}$ και $\alpha_{(2i)}$ από τον πίνακα με τα στοιχεία στην κοινόχρηστη μνήμη. Έπειτα, ο κάθε επεξεργαστής P_i αποθηκεύει το άθροισμα που υπολόγισε στη θέση α_i του πίνακα της κοινόχρηστης μνήμης, αντικαθιστώντας την αρχική τιμή της λίστας.

Σε κάθε επόμενο βήμα, ο αριθμός των επεξεργαστών που είναι σε λειτουργία μειώνονται στους μισούς. Οι επεξεργαστές αυτοί αθροίζουν τους αριθμούς που αποθηκεύτηκαν στον πίνακα της κοινόχρηστης μνήμης στο προηγούμενο βήμα με τον ίδιο τρόπο, δηλαδή ο επεξεργαστής P_i υπολογίζει το άθροισμα των $\alpha_{(2i-1)}$ και $\alpha_{(2i)}$ και το αποθηκεύει στη θέση α_i του πίνακα της κοινόχρηστης μνήμης. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να απομείνουν δύο αριθμοί, τους οποίους αθροίζει ένας επεξεργαστής και δίνει το τελικό αποτέλεσμα. Στη διαδικασία αυτή όσοι επεξεργαστές δεν είναι αναγκαίο να εργάζονται θεωρείται ότι εκτελούν no-op. Όλα τα βήματα εκτελούνται συγχρονισμένα και παράλληλα από τους επεξεργαστές.

Το μοντέλο το οποίο χρησιμοποιείται για την ανάπτυξη του αλγορίθμου είναι το EREW, αφού δεν υπάρχει ταυτόχρονη ανάγνωση ή γραφή στην κοινόχρηστη μνήμη. Συγκεκριμένα, η ανάγνωση στοιχείων από την κοινόχρηστη μνήμη γίνεται με βάση το μοναδικό αναγνωριστικό i του κάθε επεξεργαστή, ενώ το ίδιο ισχύει και για την αποθήκευση του αθροίσματος στην κοινόχρηστη μνήμη. Έτσι, αφού δεν υπάρχει ανάγνωση ή γραφή στην ίδια κυψελίδα της κοινόχρηστης μνήμης την ίδια χρονική στιγμή από διαφορετικούς επεξεργαστές, τότε χρησιμοποιείται το μοντέλο EREW.
Ακολουθεί ένα κομμάτι κώδικα, όπου γίνεται η παράλληλη άθροιση των n αριθμών με τη χρήση n/2 επεξεργαστών: [2]

```
Processors i = 1 to n/2 do in parallel
      shared array sum [1…n]
      private j, a, b
      set j = 1
      while ( i ≤ n/2^j )
            a = sum[2i - 1]
            b = sum[2i]
            sum[i]=a+b
            j = j+1
      end while
end do
```

Ο πιο πάνω αλγόριθμος εκτελείται σε χρόνο Θ(logn). Ο χρόνος αυτός οφείλεται στις Θ(logn) επαναλήψεις του βρόγχου που γίνονται παράλληλα το πολύ για n/2 επεξεργαστές. Όπως παρατηρείται, με βάση τη συνθήκη $i \leq n/2^j$ του βρόγχου, μετά από κάθε επανάληψη μειώνονται στους μισούς οι επεξεργαστές που βρίσκονται σε λειτουργία, άρα χρειάζεται χρόνος Θ(logn) για την εκτέλεση των επαναλήψεων.

Σε κάθε επανάληψη, ο κάθε επεξεργαστής διαβάζει δύο αριθμούς από την κοινόχρηστη μνήμη, υπολογίζει το άθροισμα και γράφει το αποτέλεσμα στην κοινόχρηστη μνήμη. Οι πράξεις αυτές απαιτούν σταθερό χρόνο Θ(1), άρα ο συνολικός χρόνος εκτέλεσης του αλγορίθμου παράλληλης άθροισης είναι Θ(logn) (Θ(logn) * Θ(1) = Θ(logn)).

Συνεπώς, το κόστος εκτέλεσης του αλγορίθμου είναι το γινόμενο του χρόνου εκτέλεσης με τον αριθμό των επεξεργαστών, άρα (n/2)* Θ(logn) = Θ(nlogn). Επομένως, το συνολικό κόστος εκτέλεσης του αλγορίθμου παράλληλης άθροισης είναι Θ(nlogn). Ο αλγόριθμος

αυτός δεν είναι βέλτιστος, αφού το κόστος εκτέλεσής του είναι μεγαλύτερο από το χρόνο εκτέλεσης που απαιτείται κατά τη σειριακή εκτέλεση του αλγορίθμου. Στο Κεφάλαιο 4.4 παρουσιάζουμε ένα αλγόριθμο παράλληλης άθροισης ο οποίος είναι βέλτιστος.

4.3.2 Γραφική Αναπαράσταση Αλγορίθμου

Εδώ παρουσιάζουμε την υλοποίηση της γραφικής αναπαράστασης του μη βέλτιστου αλγορίθμου Παράλληλης Άθροισης. Αρχικά έπρεπε να δημιουργηθεί η αρχική φόρμα, στην οποία ο χρήστης θα βλέπει τις επιλογές από τους αλγορίθμους που έχει, θα επιλέγει τον αλγόριθμο που επιθυμεί και στη συνέχεια ο αλγόριθμος θα εμφανίζεται πάνω σ' αυτή τη φόρμα.

Η αρχική φόρμα δημιουργήθηκε με το εργαλείο Java Swing, χρησιμοποιώντας το NetBeans IDE. Η αρχική φόρμα της εφαρμογής φαίνεται στο σχήμα που ακολουθεί.

Σχήμα 4.1 – Αρχική φόρμα της εφαρμογής

Η πιο πάνω φόρμα είναι τύπου Frame και αποτελείται από τα εξής συστατικά:

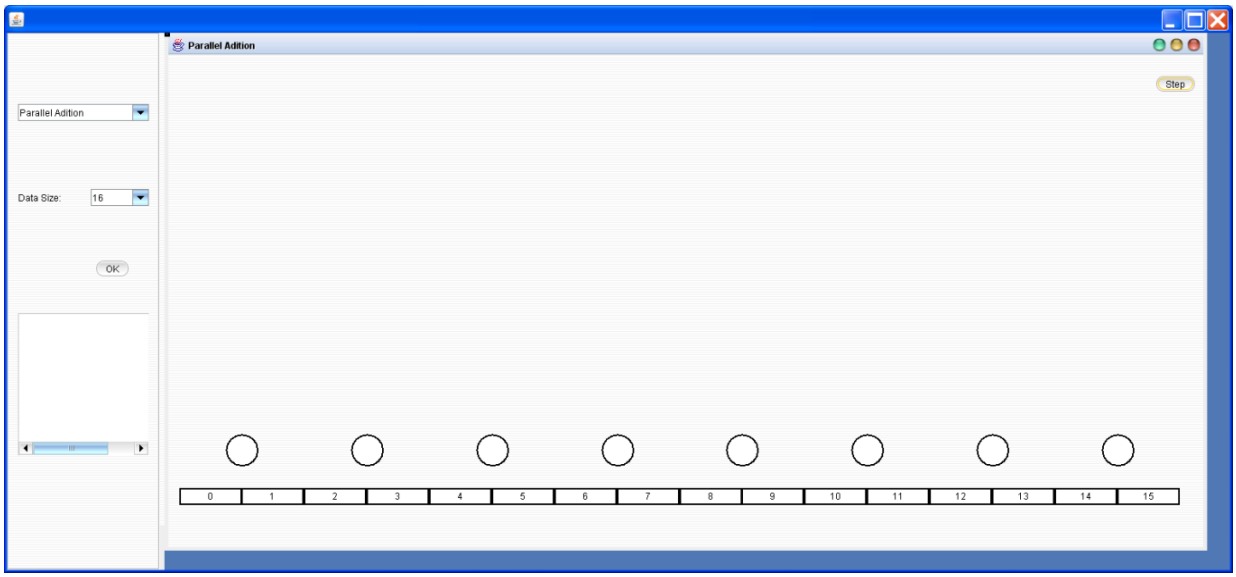
- Panel: Βρίσκεται στα αριστερά της φόρμας και περιέχει τα υπόλοιπα «μικρά» components της φόρμας, όπως συνηθίζεται, για να έχει η φόρμα μας καλύτερη αισθητική.
- Desktop Pane: Βρίσκεται στα δεξιά της φόρμας. Το Desktop Pane χρησιμοποιείται όταν θέλουμε να δημιουργήσουμε διεπαφή μεταξύ μίας φόρμας Frame και μίας

φόρμας Internal Frame. Στην περίπτωσή μας πάνω στο Desktop Pane θα τρέχουν οι γραφικές αναπαραστάσεις των αλγορίθμων οι οποίες θα είναι υλοποιημένες σε φόρμα τύπου Internal Frame.

- Combo Box: Είναι τοποθετημένα πάνω στο Panel και χρησιμοποιούνται για να εμφανίζουν στο χρήστη τις επιλογές του. Ανάλογα με τον αλγόριθμο που επιλέγει εμφανίζονται τα αντίστοιχα combo box, με τις ανάλογες επιλογές.
- Label: Είναι τοποθετημένα πάνω στο Panel και χρησιμοποιούνται για να εμφανίζουν τα ανάλογα μηνύματα στο χρήστη. Για παράδειγμα, τι αντιπροσωπεύουν οι τιμές μέσα στο κάθε combo box που υπάρχει.
- Button (OK Button): Βρίσκεται στο Panel και όταν το πατήσει ο χρήστης, εκτελείται ο κώδικας του αλγορίθμου που επεξεργάζεται τις επιλογές του χρήστη και εμφανίζεται ο αλγόριθμος που επέλεξε.
- Text Area: Βρίσκεται και αυτό στο Panel και χρησιμοποιείται για να εμφανίζονται μηνύματα προς το χρήστη.

Στη συνέχεια, ξεκίνησε η υλοποίηση της γραφικής αναπαράστασης του μη βέλτιστου αλγόριθμου παράλληλης άθροισης. Για τη δημιουργία φόρμας για την παράλληλη άθροιση, χρησιμοποιήθηκε «Internal Frame» για να μπορεί να εμφανίζεται στο «Desktop Pane» της αρχικής φόρμας. Από τα components που προσφέρει το Java Swing χρησιμοποιήθηκε μόνο ένα «Button» (OK Button), το οποίο ονομάστηκε «step» και θα είναι αυτό που στη συνέχεια θα συγχρονίζει την κίνηση στη γραφική αναπαράσταση και τα βήματα του αλγορίθμου.

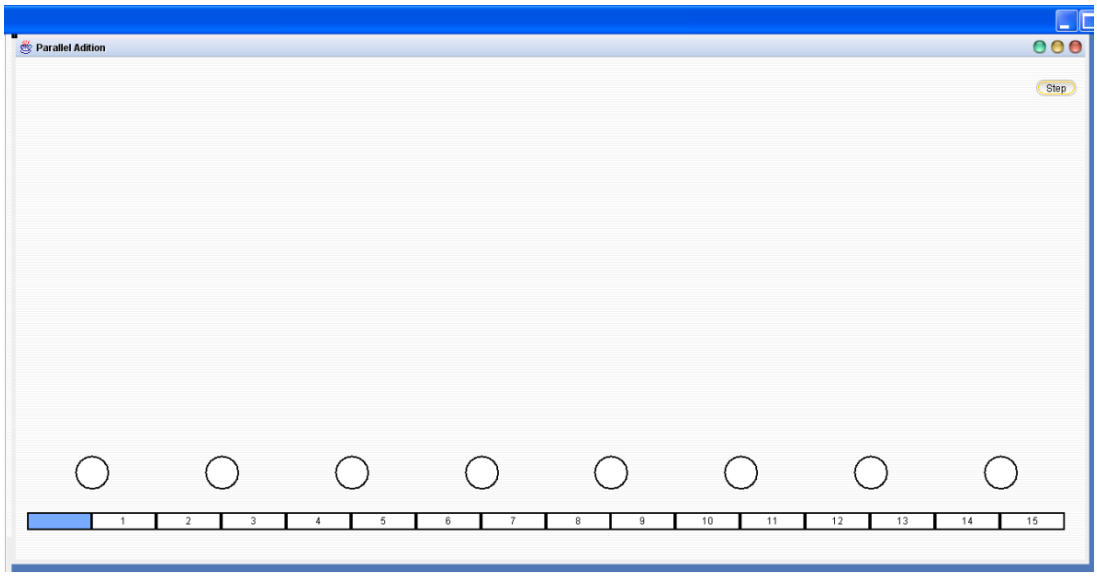
Για τη δημιουργία της γραφικής αναπαράστασης δημιουργήθηκαν κλάσεις τύπου «Processor», «MemorySlot» και «FloatingText». Η κλάση Processor δημιουργεί κύκλους που στην αναπαράστασή μας αντιπροσωπεύουν τους επεξεργαστές. Η κλάση MemorySlot δημιουργεί μικρά ορθογώνια, που αντιπροσωπεύουν τον πίνακα με τα δεδομένα μας, ενώ η κλάση FloatingText δημιουργεί συμβολοσειρές, οι οποίες θα κινούνται στους άξονες x και y, αντιπροσωπεύοντας τη μεταφορά δεδομένων προς και από τους επεξεργαστές.



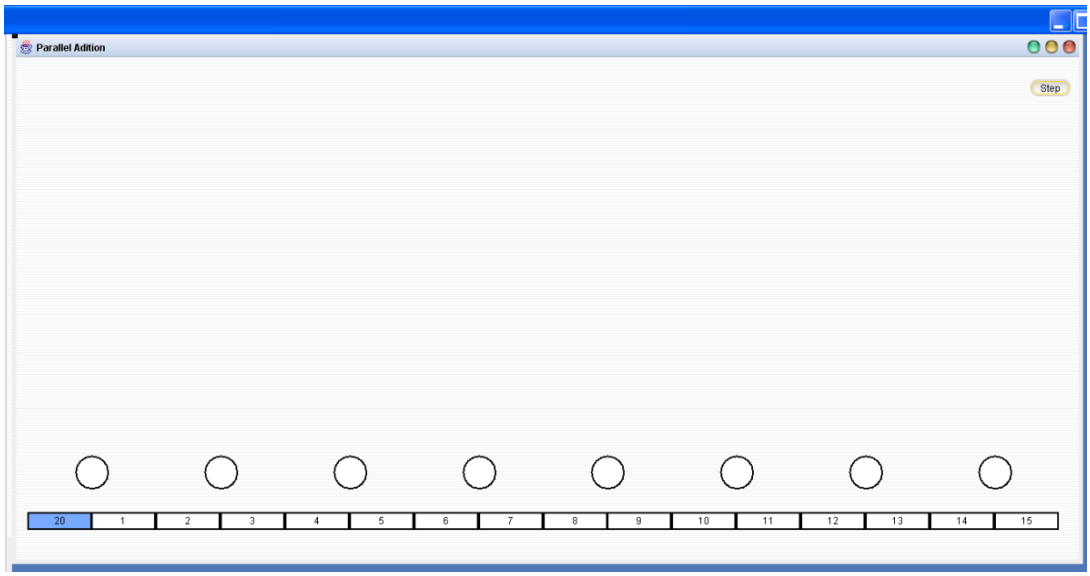
Σχήμα 4.2 – Αλγόριθμος Παράλληλης Άθροισης

Στο πιο πάνω σχήμα (Σχήμα 4.2) φαίνεται η φόρμα που εμφανίζεται, όταν ο χρήστης επιλέξει τον αλγόριθμο παράλληλης άθροισης με μέγεθος λίστας δεδομένων 16 (n=16). Επίσης φαίνεται η αναπαράσταση του πρώτου βήματος του αλγορίθμου, όπου υπάρχουν 16 στοιχεία-αριθμοί στον πίνακα (ορθογώνια) για να υπολογιστεί το άθροισμά τους και 8 επεξεργαστές (κύκλοι).

Αν παρατηρήσουμε προσεχτικά, φαίνεται ότι οι αριθμοί που βγάζει αυτόματα ο αλγόριθμος για να αθροίσει είναι σε αύξουσα σειρά από 0 μέχρι n-1 που στην περίπτωση αυτή είναι 15. Η εφαρμογή, όμως, δίνει την επιλογή στο χρήστη να τους αντικαταστήσει με αριθμούς της δικής του επιλογής, για να δημιουργήσει το δικό του σενάριο στην αρχή του αλγορίθμου πριν ξεκινήσει να εκτελείται βήμα προς βήμα. Αυτό μπορεί να το πετύχει κάνοντας κλικ πάνω στη θέση του πίνακα που θέλει να κάνει την αλλαγή. Όταν ο χρήστης πατήσει σε ένα σημείο του πίνακα, τότε αυτό επιλέγεται, αλλάζει το χρώμα της κυψελίδας από άσπρο σε γαλάζιο και ο χρήστης μπορεί, πληκτρολογώντας στο πληκτρολόγιό του τον αριθμό που θέλει, να αλλάξει την τιμή. Ακολουθεί σχετικό παράδειγμα στα σχήματα 4.3 και 4.4, όπου ο χρήστης αλλάζει την τιμή της πρώτης κυψελίδας από 0 σε 20.



Σχήμα 4.3 – Αλγόριθμος Παράλληλης Άθροισης Αλλαγή Αριθμού από τον Πίνακα (μέρος 1ο)

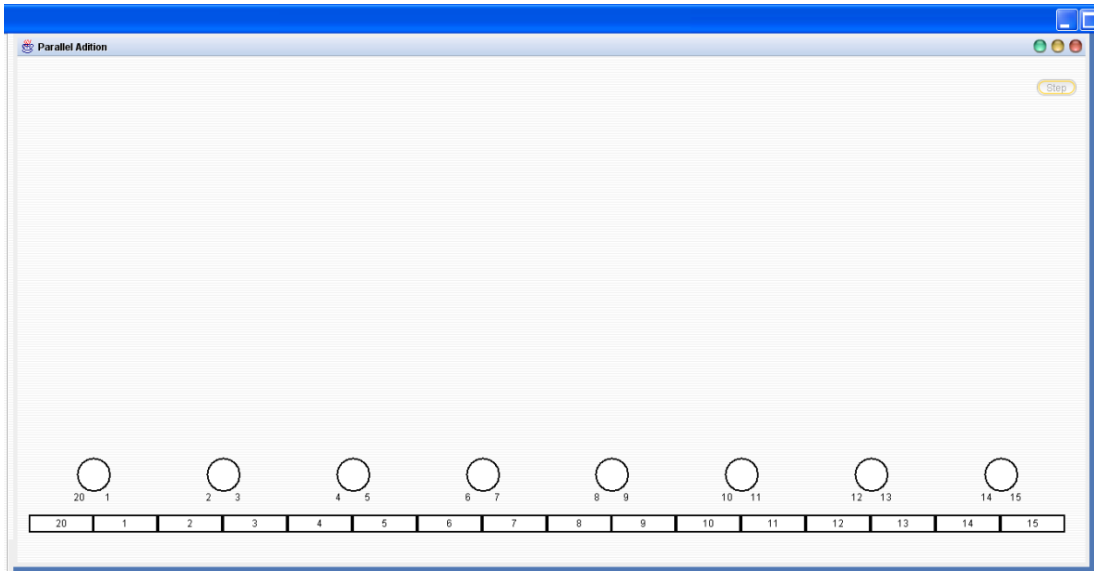


Σχήμα 4.4 – Αλγόριθμος Παράλληλης Άθροισης Αλλαγή Αριθμού από τον Πίνακα (μέρος 2ο)

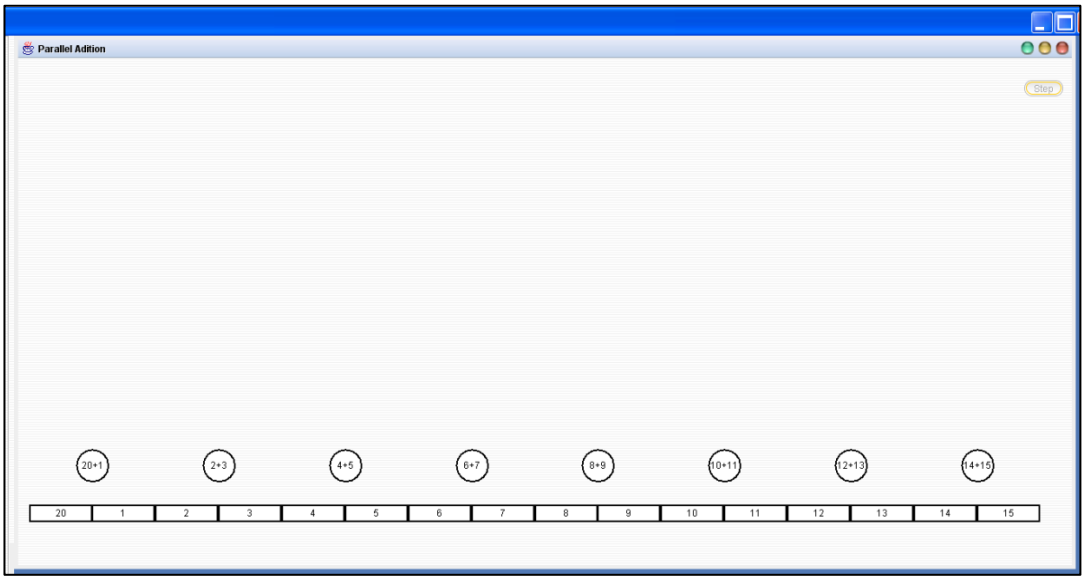
Για να πετύχουμε την αλλαγή δεδομένων με τον τρόπο αυτό, χρησιμοποιήσαμε κάποια από τα πλεονεκτήματα που προσφέρει το Java Swing, τα Event Listeners. Υπάρχουν event listeners για κάθε πιθανή πράξη και κίνηση του χρήστη πάνω στη φόρμα που θέλει η εφαρμογή να «εκμεταλλευτεί». Στη συγκεκριμένη περίπτωση, η εφαρμογή θέλει να γνωρίζει αν ο χρήστης κάνει κλικ σε οποιοδήποτε σημείο της φόρμας και αν ναι, τότε να γνωρίζει εάν στην συνέχεια πάτησε κάποιο πλήκτρο στο πληκτρολόγιο. Για το λόγο αυτό ενεργοποιήθηκαν για τη συγκεκριμένη φόρμα τα event listeners formMouseClicked και formKeyPressed.

Αφού ο χρήστης έχει στον πίνακα τους αριθμούς που θέλει να αθροίσει, τότε μπορεί να πατήσει το κουμπί step για να προχωρήσει ο αλγόριθμος στο επόμενο βήμα. Για τη λειτουργία του κουμπιού step, για τους ίδιους λόγους όπως προηγουμένως, έπρεπε να ενεργοποιηθεί το event listener btnStartActionPerformed.

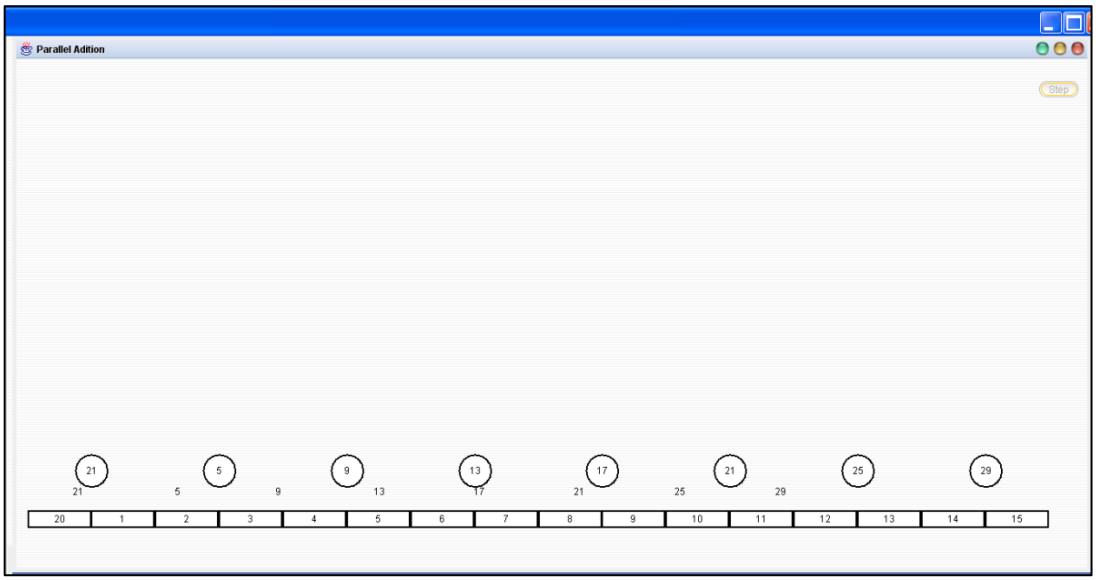
Στο επόμενο βήμα ο κάθε επεξεργαστής διαβάζει τους δύο αριθμούς που θα αθροίσει, τους αθροίζει και τους αποθηκεύει πίσω στον πίνακα. Οι θέσεις από όπου διαβάζει και γράφει ο κάθε επεξεργαστής ορίζονται σύμφωνα με τα βήματα που περιγράφηκαν στο Υποκεφάλαιο 4.3.1. Ακολουθούν σχήματα, στα οποία φαίνεται η γραφική αναπαράσταση του βήματος αυτού από την εφαρμογή.



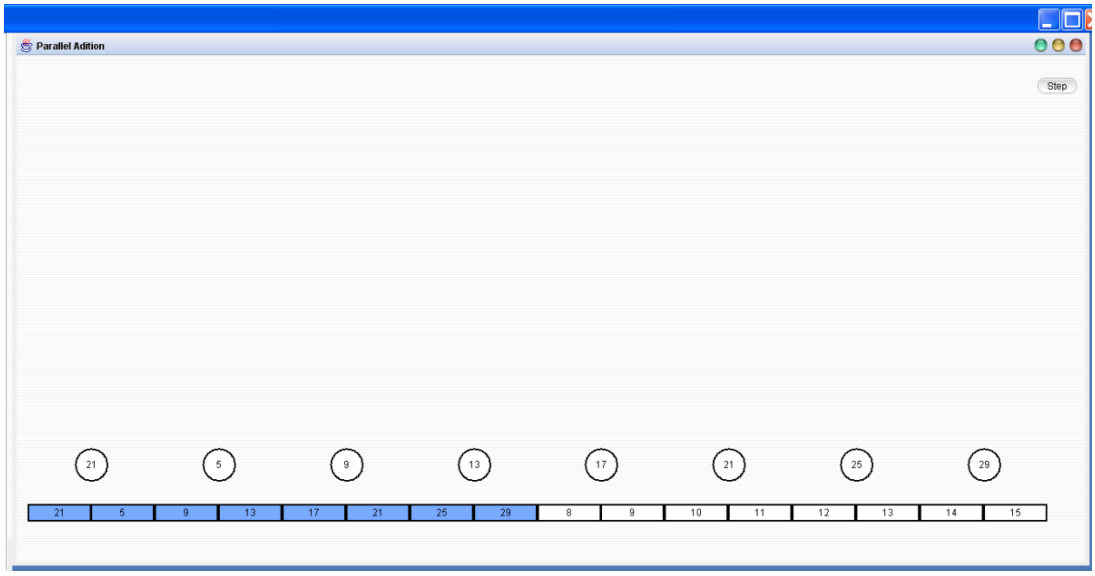
Σχήμα 4.5 – Βήμα 1^{ο},Αλγόριθμος Παράλληλης Άθροισης Μεταφορά δεδομένων στους επεξεργαστές



Σχήμα 4.6 – Βήμα 1ο,Αλγόριθμος Παράλληλης Άθροισης Άθροιση δεδομένων στους επεξεργαστές



Σχήμα 4.7 – Βήμα 1ο,Αλγόριθμος Παράλληλης Άθροισης Μεταφορά Αθροισμάτων στο Πίνακα



Σχήμα 4.8 – Βήμα 1ο,Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Αθροισμάτων στο Πίνακα

Στα πιο πάνω σχήματα (Σχήμα 4.5, 4.6, 4.7, 4.8) φαίνεται η γραφική αναπαράσταση του πρώτου βήματος του αλγορίθμου παράλληλης άθροισης. Για να είναι κατανοητό στη γραφική αναπαράσταση ποιος επεξεργαστής διαβάζει και αθροίζει ποιους αριθμούς, προστέθηκε η κίνηση των αριθμών (με τη χρήση των FloatingText που αναφέρθηκαν προηγουμένως) από τον πίνακα με τους αριθμούς προς τους επεξεργαστές και αντίστροφα, όπως φαίνεται στα σχήματα 4.5 και 4.7.

Για να γίνει αυτή η κίνηση και να φαίνεται γραφικά, χρειάστηκε να χρησιμοποιηθεί το Runnable Interface που προσφέρει η Java (υλοποιώντας το στην τρέχουσα κλάση), δηλαδή να χρησιμοποιηθούν νήματα (threads). Ο τρόπος υλοποίησης της κίνησης αυτής είναι παρόμοιος με τον τρόπο που περιγράφηκε προηγουμένως, στο Υποκεφάλαιο 3.6, στο παράδειγμα με κώδικα και σχήματα GraphicsExample.

Η διαφορά σε αυτή την περίπτωση είναι ότι η δημιουργία της κίνησης είναι λίγο πιο πολύπλοκη, γιατί υπάρχει συγκεκριμένη αρχή και τέλος για κάθε αριθμό που κινείται. Επιπλέον, οι αριθμοί δεν έχουν την ίδια απόσταση να διανύσουν αλλά πρέπει η κίνηση τους να είναι συγχρονισμένη (αφού στο μοντέλο PRAM επεξεργαστές είναι συγχρονισμένοι), δηλαδή να ξεκινούν όλοι την ίδια στιγμή και να φθάνουν όλοι την ίδια στιγμή. Για να πετύχει αυτό χρησιμοποιήθηκαν δύο μεταβλητές, «step» και «ratio». Η μεταβλητή step μετρά πόσες φορές καλείται η μέθοδος run(), ενώ η μεταβλητή ratio υπολογίζει την αναλογία σε σχέση με τα steps που έγιναν για να υπολογιστεί στη συνέχεια η απόσταση που πρέπει να διανύσει ο κάθε αριθμός σε κάθε step.

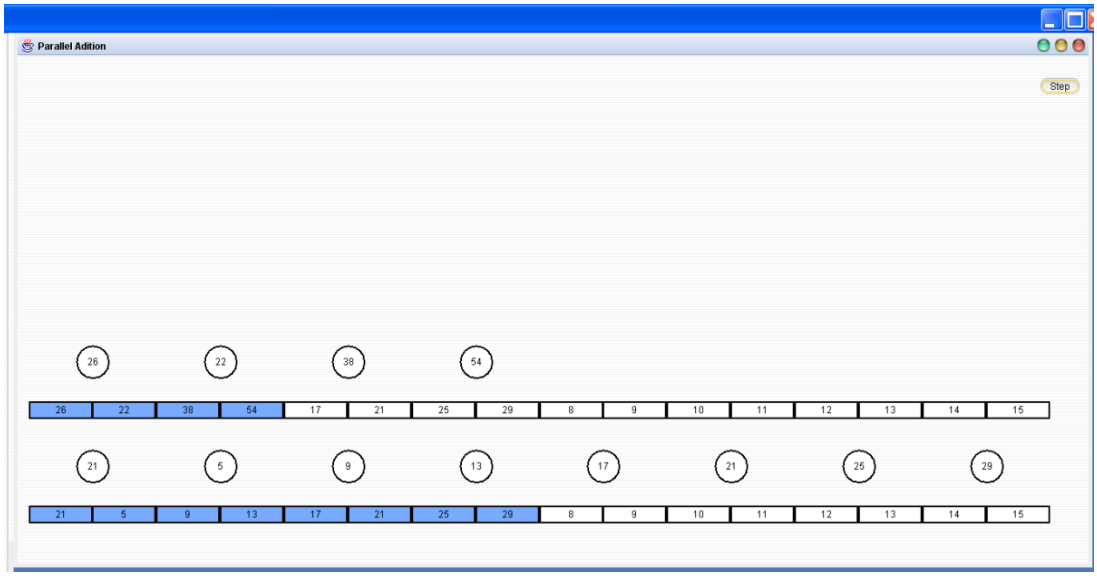
Ακολουθεί ένα μικρό κομμάτι από τον κώδικα υλοποίησης, με τις μεθόδους run() και update(…), όπου φαίνεται ο τρόπος υπολογισμού του ratio και των επόμενων συντεταγμένων στις οποίες θα βρίσκεται ο κάθε αριθμός. Η μεταβλητή int a στην μέθοδο run() είναι το ratio.

```
public void run() {
        //This function is a thread its job is to update the
        //positions of the objects and call the form's paint method
        int state = 0;
        while (runner != null) {
            //update
            if (state == 0) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 100) {
                    state = 1;
                }
                for (int i = 0; i < levels.size(); i++) {
                    levels.get(i).update(a, 2);
                }
            }
            …
        }
}
```

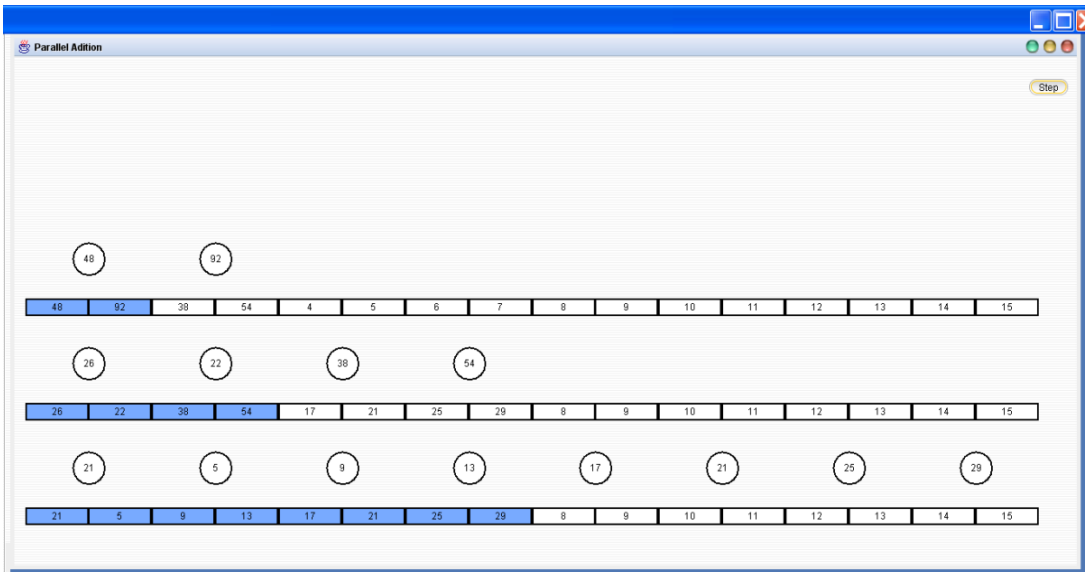
```
public void update(float ratio, int side) {
    int n = processors.size();
    if (ratio > 0.7 && animation){
        for (int i = 0; i < n; i++){
            floatingText.get(i * 2).visible = false;
            floatingText.get(i * 2 + 1).visible = false;
            processors.get(i).text=floatingText.get(i*2).text+"+"+
              floatingText.get(i*2+1).text;
            processors.get(i).numA =
              Integer.parseInt(floatingText.get(i*2).text);
            processors.get(i).numB=
              Integer.parseInt(floatingText.get(i*2+1).text);
        }
        return;
    }
    …
    for (int i = 0; i < n; i++) {
        floatingText.get(i * 2).visible = true;
        floatingText.get(i * 2).text=memorySlots.get(i * 2).text;
        floatingText.get(i*2).x =(int)(memorySlots.get(i*2).getCenterX()*(1-
             ratio)+processors.get(i).getCenterX()*ratio);
        floatingText.get(i*2).y =(int)(memorySlots.get(i*2).getCenterY()*(1-
             ratio)+processors.get(i).getCenterY()*ratio);
        floatingText.get(i*2+1).visible= true;
        floatingText.get(i*2+1).text= memorySlots.get(i*2+1).text;
        floatingText.get(i*2+1).x=(int)(memorySlots.get(i*2+1).getCenterX()*
             (1-ratio)+processors.get(i).getCenterX()*ratio);
        floatingText.get(i*2+1).y=(int)(memorySlots.get(i*2+1).getCenterY()*
             (1-ratio)+processors.get(i).getCenterY()*ratio);
    }
    …
}
```

Όταν τελειώσει ο αλγόριθμος με τη γραφική αναπαράσταση του πρώτου βήματος, τότε ενεργοποιείται και πάλι το κουμπί step και ο χρήστης μπορεί να το πατήσει για να προχωρήσει με το επόμενο βήμα. Τα βήματα που θα ακολουθήσουν κάνουν ακριβώς την ίδια διαδικασία με το πρώτο βήμα με λιγότερους όμως αριθμούς για να αθροίσουν και λιγότερους επεξεργαστές να αθροίζουν, ακριβώς όπως περιγράφεται στο Υποκεφάλαιο 4.3.1. Ο αλγόριθμος τερματίζεται όταν έχει υπολογιστεί το συνολικό άθροισμα όλων των

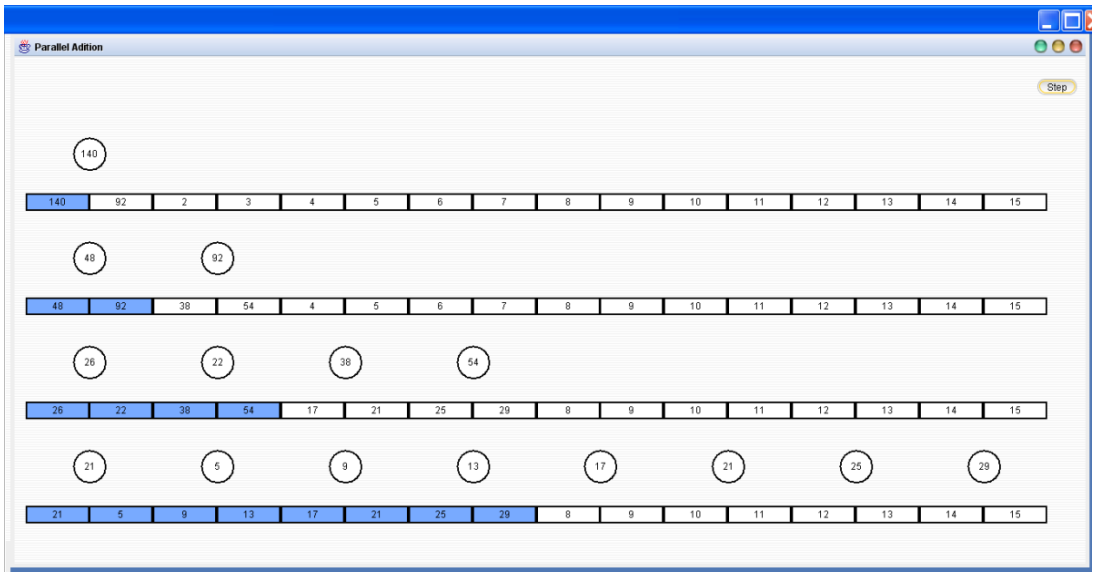
επεξεργαστών. Ακολουθούν σχήματα από το τέλος κάθε βήματος μέχρι το τέλος της άθροισης, συνεχίζοντας με το ίδιο παράδειγμα.



Σχήμα 4.9 – Βήμα 2ο,Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Αθροισμάτων στο Πίνακα



Σχήμα 4.10 – Βήμα 3ο,Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Αθροισμάτων στο Πίνακα



Σχήμα 4.11 – Βήμα 4^{o},Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Τελικού Αθροίσματος στο Πίνακα

Όπως παρατηρείται στο πιο πάνω σχήμα (Σχήμα 4.11), το αποτέλεσμα της παράλληλης άθροισης του παραδείγματος αυτού είναι 140. Ο πίνακας αρχικά περιείχε τις τιμές 20, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15. Κάνοντας την άθροιση των αριθμών αυτών σειριακά το αποτέλεσμα είναι 140, όπως και το αποτέλεσμα που έδωσε στο τέλος η γραφική αναπαράσταση.

4.4 Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης

4.4.1 Περιγραφή Αλγορίθμου

Για να έχουμε ένα βέλτιστο παράλληλο αλγόριθμο άθροισης θα πρέπει να τροποποιηθεί ο αλγόριθμος παράλληλης άθροισης για να μειωθεί το κόστος. Για να μειωθεί το κόστος θα πρέπει να μειωθούν οι επεξεργαστές. Έτσι, για την επίλυση του προβλήματος άθροισης με βέλτιστο τρόπο χρησιμοποιούνται n/logn επεξεργαστές [2].

Ο αλγόριθμος χωρίζεται σε δύο φάσεις. Στην πρώτη φάση, ο κάθε ένας από τους n/logn επεξεργαστές εκτελεί σειριακό άθροισμα σε ένα σύνολο στοιχείων της κοινόχρηστης μνήμης που του αντιστοιχούν. Στη δεύτερη φάση, εκτελείται κανονικά η παράλληλη άθροιση που περιγράφηκε στο Υποκεφάλαιο 4.3.1 με (n/logn)/2 επεξεργαστές και n/logn στοιχεία.

Στην πρώτη φάση, οι n/logn επεξεργαστές λειτουργούν παράλληλα και ο καθένας εκτελεί σειριακό άθροισμα για logn διαφορετικά στοιχεία από την κοινόχρηστη μνήμη.

Συγκεκριμένα, ο πρώτος επεξεργαστής αθροίζει το πρώτο σύνολο από logn στοιχεία της κοινόχρηστης μνήμης, ο δεύτερος το δεύτερο σύνολο από logn στοιχεία κ.ο.κ. μέχρι τον n/logn επεξεργαστή που αθροίζει το τελευταίο σύνολο από logn στοιχεία. Μόλις ολοκληρωθεί η σειριακή άθροιση, ο κάθε επεξεργαστής αποθηκεύει το άθροισμα που υπολόγισε στη θέση της κοινόχρηστης μνήμης που του αντιστοιχεί, δηλαδή στη θέση με αριθμό το αναγνωριστικό του επεξεργαστή. Έτσι, απομένουν n/logn στοιχεία για τα οποία πρέπει να υπολογιστεί το άθροισμα.

Στη δεύτερη φάση, χρησιμοποιούνται (n/logn)/2 επεξεργαστές για την εκτέλεση του παράλληλου αλγορίθμου, αφού το σύνολο των στοιχείων που πρέπει να αθροιστούν είναι n/logn. Η διαδικασία άθροισης είναι ακριβώς η ίδια με αυτή που περιγράφηκε στο Υποκεφάλαιο 4.3.1.

Το μοντέλο το οποίο χρησιμοποιείται για την ανάπτυξη του αλγορίθμου είναι το EREW, αφού δεν υπάρχει ταυτόχρονη ανάγνωση ή γραφή στην κοινόχρηστη μνήμη. Συγκεκριμένα, η ανάγνωση στοιχείων από την κοινόχρηστη μνήμη γίνεται βάση του μοναδικού αναγνωριστικού i του κάθε επεξεργαστή, ενώ το ίδιο ισχύει και για την αποθήκευση του αθροίσματος στην κοινόχρηστη μνήμη. Έτσι, αφού δεν υπάρχει ανάγνωση ή γραφή στην ίδια κυψελίδα της κοινόχρηστης μνήμης την ίδια χρονική στιγμή από διαφορετικούς επεξεργαστές, τότε χρησιμοποιείται το μοντέλο EREW.

Ο χρόνος εκτέλεσης της πρώτης φάσης του αλγορίθμου είναι Θ(logn) και αυτό οφείλεται στο σειριακό άθροισμα των logn στοιχείων που κάνει ο κάθε επεξεργαστής παράλληλα. Ο χρόνος εκτέλεσης για τη δεύτερη φάση είναι Θ(log(n/logn)). Αφού στη δεύτερη φάση χρησιμοποιείται ο αλγόριθμος παράλληλης άθροισης που, όπως περιγράφηκε προηγουμένως, εκτελείται σε χρόνο Θ(logn) για n στοιχεία, για n/logn στοιχεία ο χρόνος εκτέλεσης είναι Θ(log(n/logn)).

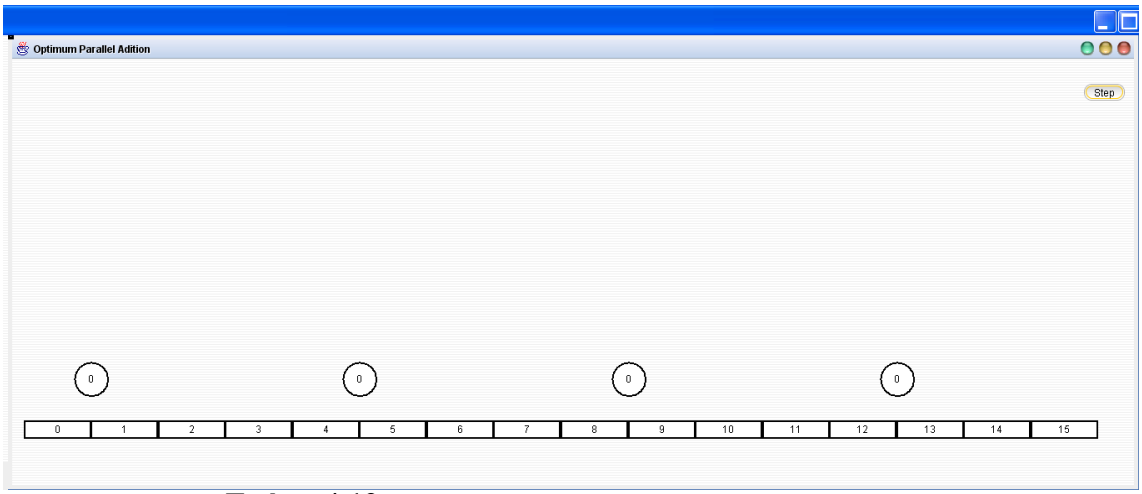
Προφανώς, αφού ο χρόνος εκτέλεσης της πρώτης φάσης είναι μεγαλύτερος από αυτόν της δεύτερης, τότε ο συνολικός χρόνος εκτέλεσης του αλγορίθμου είναι Θ(logn). Συνεπώς, το κόστος του βέλτιστου αλγορίθμου άθροισης είναι το γινόμενο του χρόνου εκτέλεσης και των επεξεργαστών, δηλαδή Θ(logn) * n/logn = Θ(n). Μπορεί ο χρόνος εκτέλεσης του βέλτιστου αλγόριθμου να παραμένει ο ίδιος με τον αρχικό αλγόριθμο, αλλά το κόστος έχει μειωθεί.

Επομένως, ο βέλτιστος αλγόριθμος παράλληλης άθροισης υπολογίζει το άθροισμα n αριθμών με n/logn επεξεργαστές σε ασυμπτωτικό χρόνο Θ(logn), ίδιο με τον αλγόριθμο παράλληλης άθροισης και με κόστος της τάξεως του χρόνου του καλύτερου σειριακού αλγορίθμου άθροισης Θ(n).

4.4.2 Γραφική Αναπαράσταση Αλγορίθμου

Εδώ παρουσιάζουμε την υλοποίηση της γραφικής αναπαράστασης του βέλτιστου αλγορίθμου Παράλληλης Άθροισης. Όπως έχει αναφερθεί, ο βέλτιστος αλγόριθμος παράλληλης άθροισης χωρίζεται σε δύο φάσεις. Η δεύτερη φάση του αλγορίθμου αυτού είναι η εκτέλεση του μη-βέλτιστου αλγορίθμου παράλληλης άθροισης, του οποίου η περιγραφή έγινε στο Υποκεφάλαιο 4.3.1 και η υλοποίηση αναλύθηκε στο Υποκεφάλαιο 4.3.2.

Επομένως, στο υποκεφάλαιο αυτό αρχικά θα αναλυθεί η υλοποίηση της γραφικής αναπαράστασης της πρώτης φάσης του βέλτιστου αλγορίθμου παράλληλης άθροισης. Είναι σημαντικό να αναφερθεί ότι και σε αυτό τον αλγόριθμο χρησιμοποιήθηκε «Internal Frame» για τη δημιουργία της φόρμας, για να μπορεί να εμφανίζεται στο «Desktop Pane» της αρχικής φόρμας. Επίσης, από τα components που προσφέρει το Java Swing χρησιμοποιήθηκε μόνο ένα «Button» (OK Button), το οποίο ονομάστηκε «step» και θα είναι αυτό που στη συνέχεια θα συγχρονίζει την κίνηση στη γραφική αναπαράσταση και τα βήματα του αλγορίθμου.Για τη δημιουργία της γραφικής αναπαράστασης χρησιμοποιήθηκαν οι ίδιες κλάσεις που είχαν δημιουργηθεί για το μη-βέλτιστο αλγόριθμο παράλληλης άθροισης, «Processor», «MemorySlot» και «FloatingText», οι οποίες αναφέρθηκαν στο Υποκεφάλαιο 4.3.2.



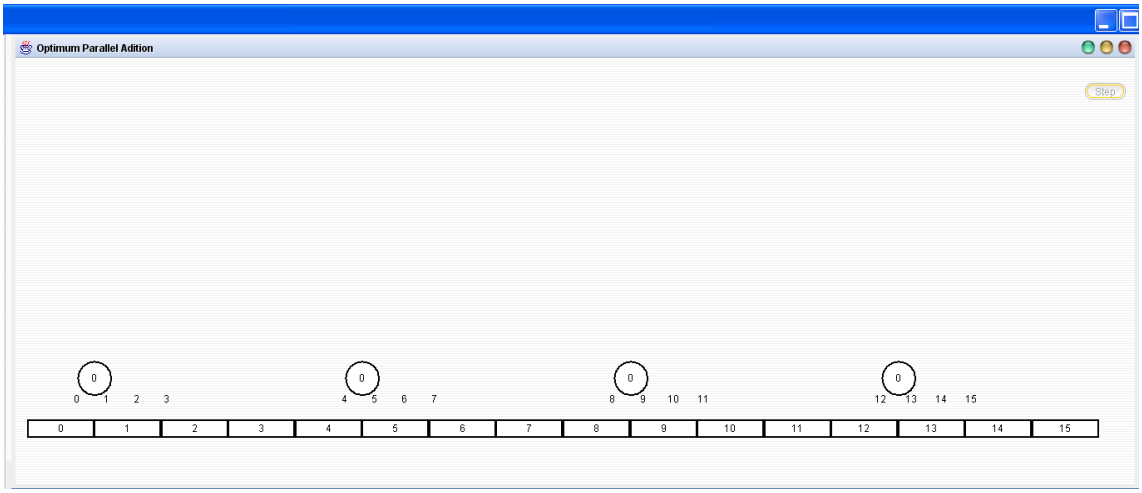
Σχήμα 4.12 – Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης

Στο πιο πάνω σχήμα (Σχήμα 4.12), φαίνεται η φόρμα που εμφανίζεται, όταν ο χρήστης επιλέξει το βέλτιστο αλγόριθμο παράλληλης άθροισης με μέγεθος λίστας δεδομένων 16 (n=16). Επίσης, φαίνεται η αναπαράσταση του πρώτου βήματος του αλγορίθμου, όπου υπάρχουν 16 στοιχεία-αριθμοί στο πίνακα (ορθογώνια) για να υπολογιστεί το άθροισμά τους και 4 επεξεργαστές (κύκλοι).

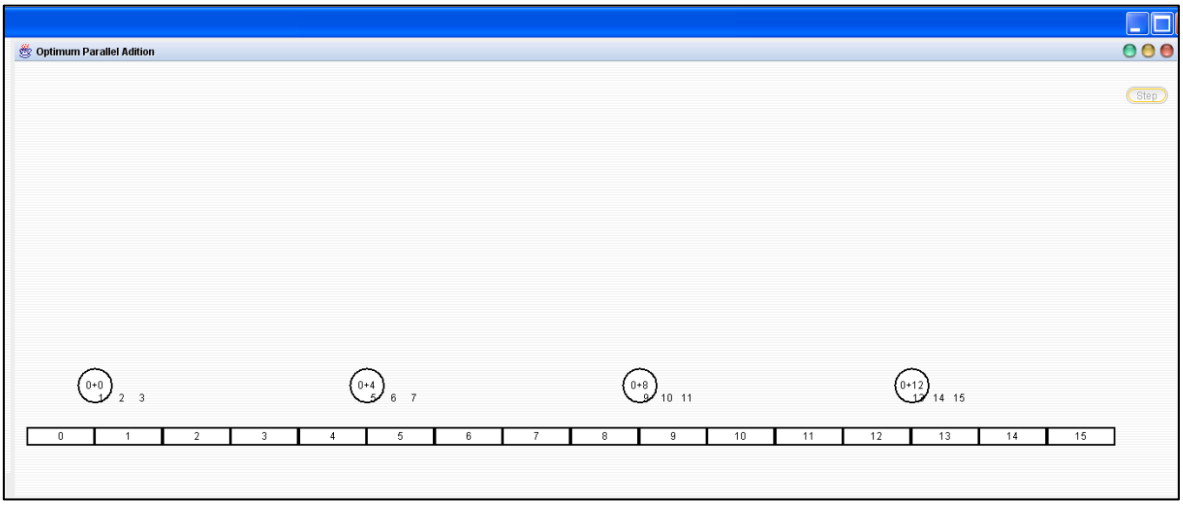
Αν παρατηρήσουμε προσεχτικά, οι αριθμοί που βγάζει αυτόματα ο αλγόριθμος για να αθροίσει είναι σε αύξουσα σειρά από 0 μέχρι n-1 που στην περίπτωση αυτή είναι 15, όπως ήταν και στο μη βέλτιστο αλγόριθμο παράλληλης άθροισης στο Υποκεφάλαιο 4.3.2. Και πάλι, η εφαρμογή δίνει την επιλογή στο χρήστη να τους αντικαταστήσει με αριθμούς της δικής του επιλογής, για να δημιουργήσει το δικό του σενάριο στην αρχή του αλγορίθμου, πριν ξεκινήσει να εκτελείται βήμα προς βήμα. Ο τρόπος με τον οποίο μπορεί να το πετύχει αυτό ο χρήστης έχει περιγραφεί στο Υποκεφάλαιο 4.3.2 μαζί με παραδείγματα, όπως έχει περιγραφεί και ο τρόπος υλοποίησής του στο Java Swing.

Ο χρήστης, αφού έβαλε τους αριθμούς που θέλει να αθροίσει στον πίνακα, μπορεί να πατήσει το κουμπί step για να ξεκινήσει ο αλγόριθμος να εκτελεί την πρώτη φάση. Στην πρώτη φάση, ο κάθε επεξεργαστής διαβάζει τους αριθμούς που του αντιστοιχούν, για να τους αθροίσει, τους αθροίζει και τους αποθηκεύει πίσω στον πίνακα. Οι θέσεις από όπου διαβάζει και γράφει ο κάθε επεξεργαστής ορίζονται σύμφωνα με τα βήματα που περιγράφηκαν στο Υποκεφάλαιο 4.4.1.

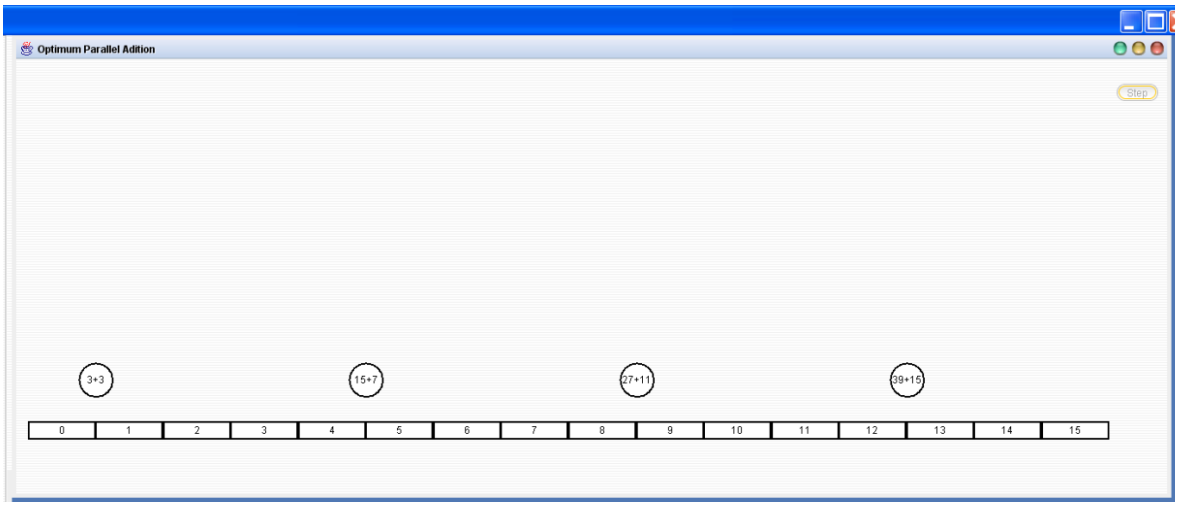
Η διαφορά όμως του αλγορίθμου αυτού από το μη βέλτιστο αλγόριθμο παράλληλης άθροισης είναι ότι οι αριθμοί που πρέπει να αθροίσει ο κάθε επεξεργαστής σε αυτή τη φάση εξαρτώνται από το n και δεν είναι πάντοτε 2, αλλά logn. Σύμφωνα με το προηγούμενο παράδειγμα από το σχήμα 4.12, αφού n=16, τότε ο κάθε επεξεργαστής έχει να αθροίσει 4 αριθμούς στην πρώτη φάση. Αν και η άθροιση αυτών των αριθμών γίνεται σειριακά από κάθε επεξεργαστή, γίνεται από όλους τους επεξεργαστές παράλληλα. Ακολουθούν σχήματα, στα οποία φαίνεται η γραφική αναπαράσταση της πρώτης φάσης του βέλτιστου αλγορίθμου παράλληλης άθροισης από την εφαρμογή.



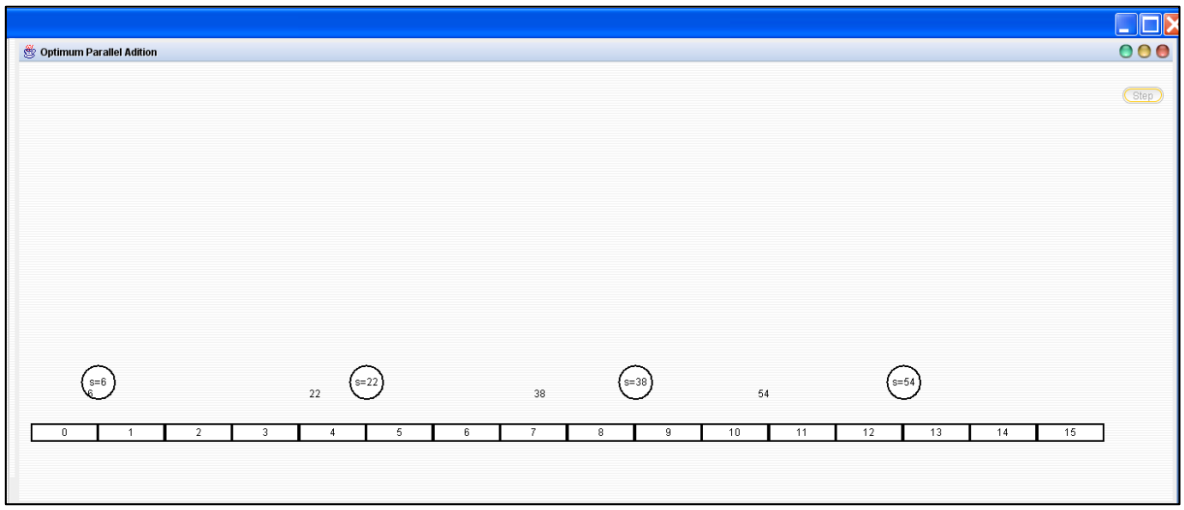
Σχήμα 4.13 – Φάση 1η, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Μεταφορά δεδομένων στους επεξεργαστές



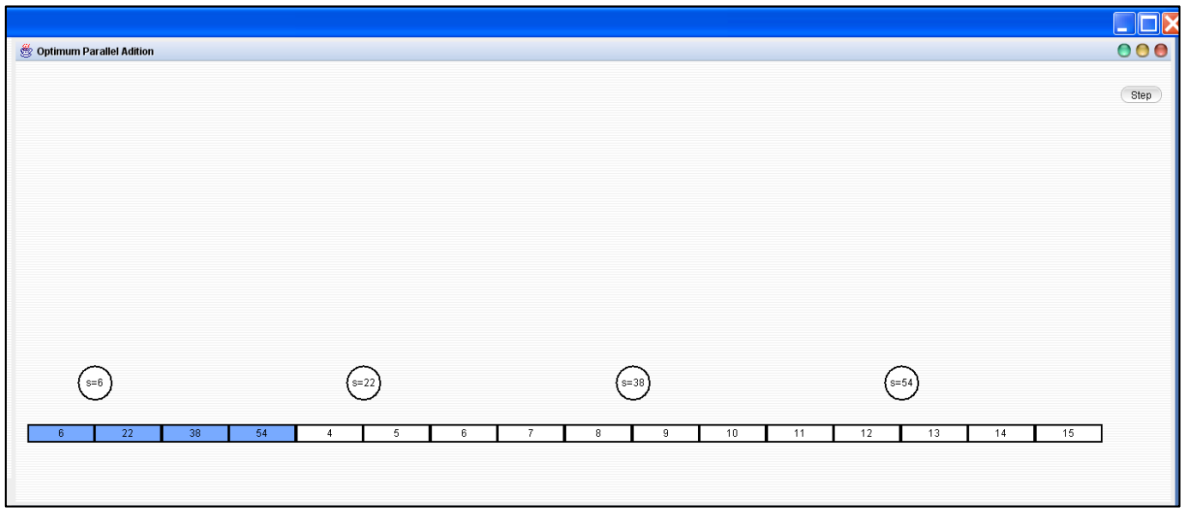
Σχήμα 4.14 – Φάση 1η, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Αρχή Σειριακής Άθροισης δεδομένων στους επεξεργαστές



Σχήμα 4.15 – Φάση 1η, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Τέλος Σειριακής Άθροισης δεδομένων στους επεξεργαστές



Σχήμα 4.16 – Φάση 1η, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Μεταφορά Αθροισμάτων στο Πίνακα

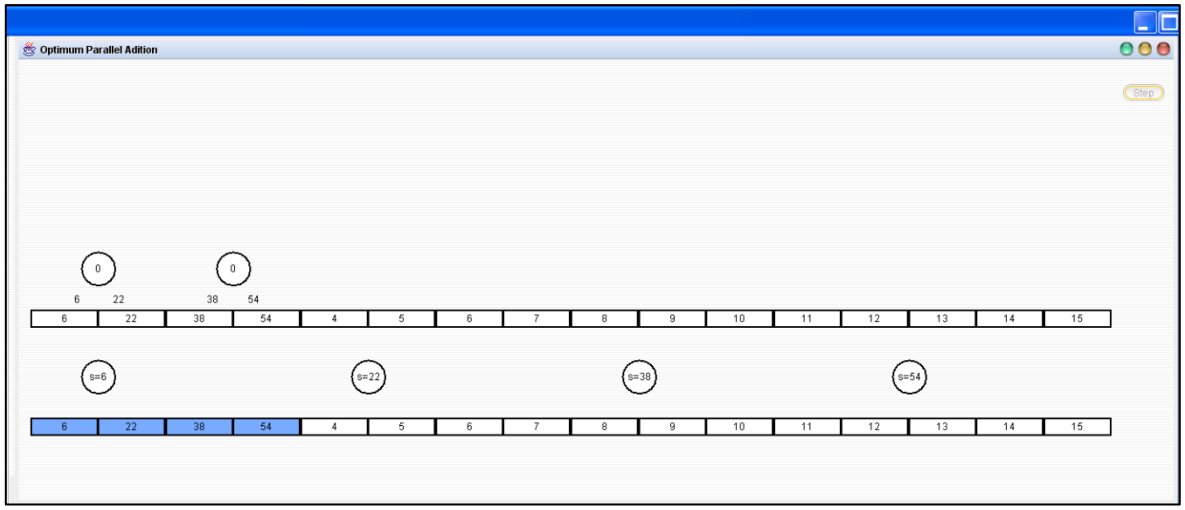


Σχήμα 4.17 – Φάση 1η, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Αθροισμάτων στο Πίνακα

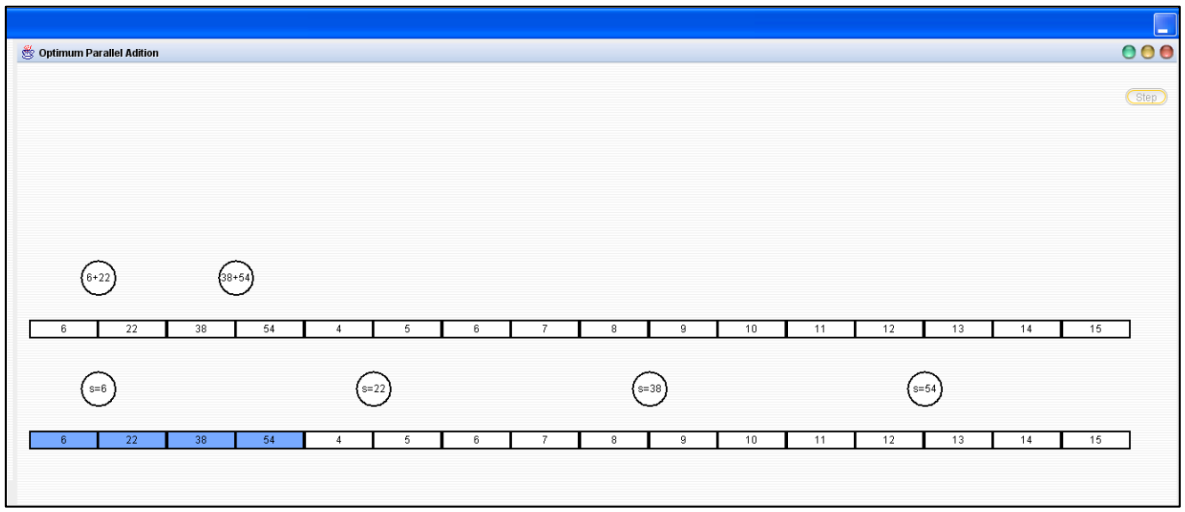
Στα πιο πάνω σχήματα (Σχήμα 4.13, 4.14, 4.15, 4.16, 4.17) φαίνεται η γραφική αναπαράσταση της πρώτης φάσης του βέλτιστου αλγορίθμου παράλληλης άθροισης. Όπως περιγράφηκε και εξηγήθηκε στο Υποκεφάλαιο 4.3.1, με τον ίδιο τρόπο υλοποιήθηκε η κίνηση των αριθμών (FloatingText) και σε αυτή την γραφική αναπαράσταση.

Όταν τελειώσει ο αλγόριθμος με τη γραφική αναπαράσταση της πρώτης φάσης, ενεργοποιείται και πάλι το κουμπί step και ο χρήστης μπορεί να το πατήσει για να προχωρήσει με την επόμενη φάση. Η επόμενη φάση, όπως αναφέρθηκε και προηγουμένως, αποτελείται από το μη βέλτιστο αλγόριθμο παράλληλης άθροισης και τα βήματα που τον απαρτίζουν.

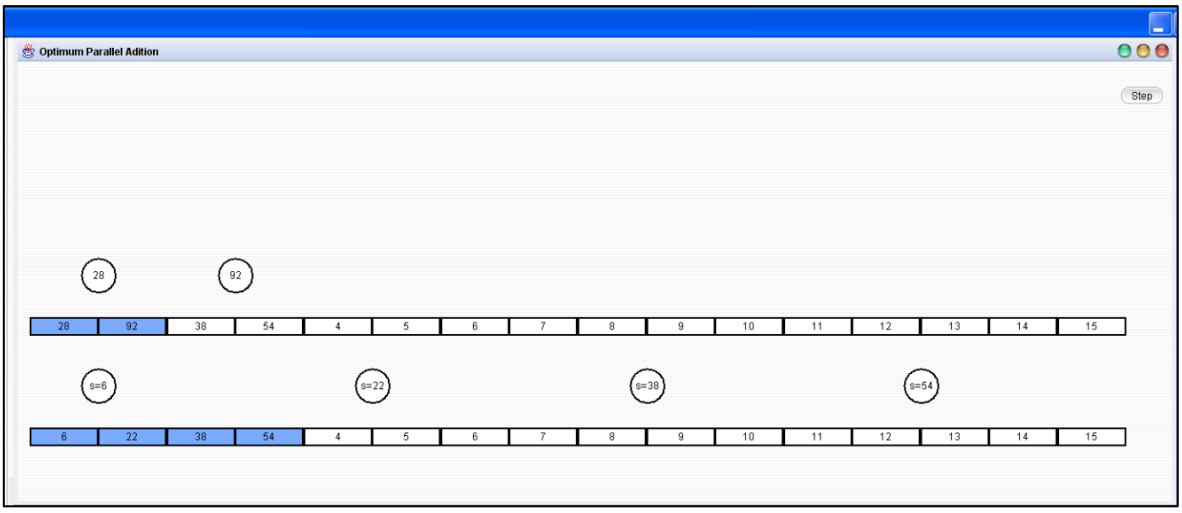
Συνεπώς, στη φάση αυτή εκτελείται ο μη βέλτιστος αλγόριθμος παράλληλης άθροισης αλλά αντί με n στοιχεία και n/2 επεξεργαστές, όπως στα υποκεφάλαια 4.3.1 και 4.3.2, έχει n/logn στοιχεία για να αθροίσει με (n/logn)/2 επεξεργαστές. Ο αλγόριθμος τερματίζεται όταν έχει υπολογιστεί το συνολικό άθροισμα όλων των επεξεργαστών. Ακολουθούν σχήματα από το τέλος κάθε βήματος της δεύτερης φάσης μέχρι το τέλος της άθροισης, συνεχίζοντας με το ίδιο παράδειγμα.



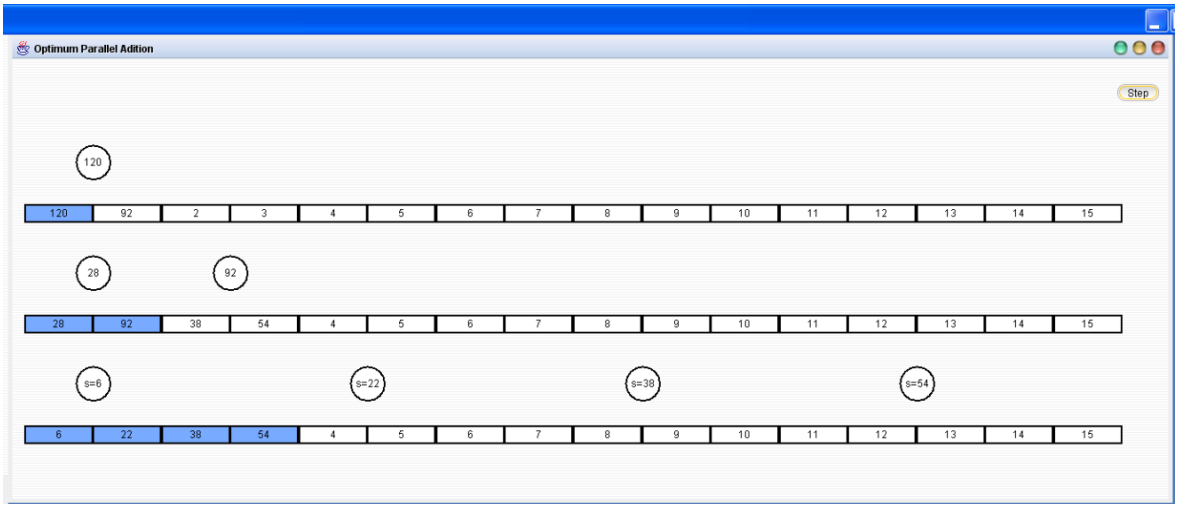
Σχήμα 4.18 – Φάση 2η-Βήμα 1ο, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Μεταφορά δεδομένων στους επεξεργαστές



Σχήμα 4.19 – Φάση 2η-Βήμα 1ο, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Άθροιση δεδομένων στους επεξεργαστές



Σχήμα 4.20 – Φάση 2η-Βήμα 1ο, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Αθροισμάτων στο Πίνακα



Σχήμα 4.21 – Φάση 2η-Βήμα 2ο, Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Αποθήκευση Τελικού Αθροίσματος στο Πίνακα

Όπως παρατηρούμε στο πιο πάνω σχήμα (Σχήμα 4.21), το αποτέλεσμα της βέλτιστης παράλληλης άθροισης του παραδείγματος αυτού είναι 120. Ο πίνακας αρχικά περιείχε τις τιμές 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15. Κάνοντας την άθροιση αυτών των αριθμών σειριακά το αποτέλεσμα είναι 120, όπως και το αποτέλεσμα που έδωσε στο τέλος η γραφική αναπαράσταση.

Κεφάλαιο 5

Αλγόριθμος W

5.1 Πρόβλημα Write-All στο F-PRAM με n επεξεργαστές

Στο υποκεφάλαιο αυτό θα μελετηθεί το πρόβλημα Write-All [6] στο μοντέλο F-PRAM [2,6] που, όπως έχει αναφερθεί προηγουμένως (Υποκεφάλαιο 2.3), είναι ένα συγχρονισμένο μοντέλο με πρόσβαση στην κοινόχρηστη μνήμη, του οποίου οι επεξεργαστές υπόκεινται σε σφάλματα κατάρρευσης. Το πρόβλημα Write-All αναφέρεται στην περίπτωση κατά την οποία, δεδομένης μίας λίστας n στοιχείων, όπου αρχικά όλα τα στοιχεία έχουν την τιμή 0, ζητείται να μετατραπούν οι τιμές των στοιχείων σε 1 χρησιμοποιώντας p επεξεργαστές στο μοντέλο F-PRAM. Δηλαδή, κατά την επίλυση του προβλήματος μπορούν να καταρρεύσουν μέχρι f<p επεξεργαστές [2].

5.2 Αλγόριθμος W

5.2.1 Περιγραφή Αλγορίθμου

Ο Αλγόριθμος W [2,6] αποτελείται από τέσσερις φάσεις και από δύο διανύσματα, τα οποία αναπαριστούν νοητά δυαδικά δέντρα. Συγκεκριμένα, ο αλγόριθμος αυτός εκτελεί ένα βρόγχο ο οποίος αποτελείται από τις τέσσερις φάσεις που αναφέρθηκαν μέχρι να λυθεί το πρόβλημα.

Στην πρώτη φάση, ονομάστε την W1, γίνεται καταμέτρηση των επεξεργαστών για εύρεση σφαλμάτων. Χρησιμοποιείται ένα διάνυσμα δ_1 που αναπαρίσταται ως ένα νοητό δυαδικό δέντρο, *δέντρο καταμέτρησης*, για την καταμέτρηση των επεξεργαστών που δεν έχουν καταρρεύσει. Οι επεξεργαστές, ξεκινώντας από τα φύλλα του δέντρου, στα οποία τοποθετούνται με βάση τον προσωπικό τους αριθμό, γράφουν την τιμή 1 και ανεβαίνουν στο δέντρο τρέχοντας μια εκδοχή του αλγορίθμου παράλληλης άθροισης στο PRAM. Τέλος, στη ρίζα του δέντρου αποθηκεύεται ο υπολογισμένος αριθμός των επεξεργαστών που δεν έχουν καταρρεύσει. Αυτός ο αριθμός είναι μια *υπέρ-εκτίμηση* του πραγματικού αριθμού των επεξεργαστών που δεν έχουν καταρρεύσει ακόμη[2,6].

Στη δεύτερη φάση, ονομάστε την W2, γίνεται μια ισοζυγισμένη ανάθεση επεξεργαστών σε στοιχεία της λίστας για να γίνει ανοχή των σφαλμάτων. Χρησιμοποιείται ένα διάνυσμα δ_2 που αναπαρίσταται ως ένα νοητό δυαδικό δέντρο, *δέντρο προόδου*, για την ανάθεση επεξεργαστών σε στοιχεία της λίστας. Σ' αυτή τη φάση οι επεξεργαστές ξεκινούν από τη ρίζα του δέντρου και κατεβαίνουν προς τα κάτω. Στη ρίζα του δέντρου υπάρχει μία εκτίμηση του αριθμού των στοιχείων της λίστας που έχουν την τιμή 1 (η οποία υπολογίστηκε προηγουμένως στην φάση W4, η οποία περιγράφεται πιο κάτω) και έτσι, ανάλογα με τον αριθμό των υπολειπόμενων μηδενικών τις λίστας, οι επεξεργαστές διαχωρίζονται στα φύλλα του δέντρου. Με αυτόν τον τρόπο μπορούμε να πούμε ότι οι επεξεργαστές ανατίθενται (ισοζυγισμένα) στα φύλλα του δέντρου που βρίσκονται τα στοιχεία της λίστας.

Η τρίτη φάση, ονομάστε την W3, ονομάζεται και φάση εργασίας για το λόγο ότι είναι η φάση στην οποία οι ενεργοί επεξεργαστές βρίσκονται στα φύλλα του δέντρου προόδου, δ_2, (μετά από τη φάση W2) και αλλάζουν την τιμή του στοιχείου που αντιστοιχεί στο φύλλο που βρίσκονται από 0 σε 1.

Τέλος, στην τέταρτη φάση, ονομάστε την W4, γίνεται η αξιολόγηση της προόδου. Οι επεξεργαστές ξεκινούν από τα φύλλα του δέντρου προόδου, δ_2, όπου βρέθηκαν από την προηγούμενη φάση, W3, και ανεβαίνουν προς τα πάνω εκτελώντας μια εκδοχή του αλγόριθμου παράλληλης άθροισης του PRAM, για να υπολογίσουν τον αριθμό των στοιχείων της λίστας που έχουν τιμή 1. Ο συνολικός αριθμός των στοιχείων που έχουν την τιμή 1, βρίσκεται αποθηκευμένος στη ρίζα του δέντρου και είναι μία *υποεκτίμηση* του πραγματικού αριθμού των στοιχείων που έχουν τιμή 1[2,6].

Η εκτέλεση του βρόγχου των τεσσάρων φάσεων τερματίζεται μόνο όταν η ρίζα του δέντρου προόδου, δ_2, είναι ίση με το n. Ακολουθεί ένα κομμάτι κώδικα για τον αλγόριθμο W βάση των τεσσάρων φάσεων που περιγράφηκαν παραπάνω [2,6]:

```
Processors i=1,…,n do in parallel
      Phase W3
      Phase W4
      while δ2[1] ≠ n do
            Phase W1
            Phase W2
            Phase W3
            Phase W4
      end while
end do
```

5.2.2 Γραφική Αναπαράσταση Αλγορίθμου

Μετά την υλοποίηση του μη βέλτιστου και του βέλτιστου αλγορίθμου παράλληλης άθροισης, εδώ παρουσιάζουμε την υλοποίηση της γραφικής αναπαράστασης του Αλγορίθμου W.

Όπως και για τους αλγόριθμους παράλληλης άθροισης, χρησιμοποιήθηκε και πάλι «Internal Frame» για τη δημιουργία της φόρμας πάνω στην οποία θα τρέχει η γραφική αναπαράσταση του αλγορίθμου, για να μπορεί να εμφανίζεται στο «Desktop Pane» της αρχικής φόρμας της εφαρμογής. Από τα components που προσφέρει το Java Swing χρησιμοποιήθηκε ένα «Button» (OK Button), το οποίο ονομάστηκε «step» και θα είναι αυτό που θα συγχρονίζει την κίνηση στη γραφική αναπαράσταση και τα βήματα του αλγορίθμου. Επίσης χρησιμοποιήθηκαν τρία «Labels» στα οποία εμφανίζονται διάφορα μηνύματα για να ενημερώνεται ο χρήστης για θέματα όπως για παράδειγμα σε ποια φάση βρίσκεται τη συγκεκριμένη στιγμή ο αλγόριθμος και σε ποιο δέντρο εργάζεται.

Σύμφωνα με την περιγραφή του Αλγορίθμου W, ο αλγόριθμος χωρίζεται σε τέσσερις φάσεις, W1, W2, W3, W4, και χρησιμοποιεί δύο διανύσματα, τα οποία αναπαριστούν νοητά δυαδικά δέντρα. Έτσι, για την υλοποίηση της γραφικής αναπαράστασης του αλγορίθμου αυτού, χρησιμοποιήθηκαν δύο πίνακες για την αποθήκευση των διανυσμάτων. Επίσης, σχεδιάστηκαν με τη χρήση της κλάσης Graphics δύο διαφορετικά δέντρα, προόδου και καταμέτρησης.

Μόλις ξεκινήσει η εκτέλεση του αλγορίθμου, πριν εισέλθει στο βρόγχο, εκτελούνται πρώτα οι φάσεις W3 και W4, σύμφωνα με το κομμάτι κώδικα του Αλγόριθμου W που παρουσιάσαμε στο προηγούμενο Υποκεφάλαιο 5.2.1. Έτσι, η υλοποίηση της γραφικής αναπαράστασης του αλγορίθμου W ξεκίνησε με την υλοποίηση της φάσης W3 για να συνεχίσει με τις φάσεις W4, W1 και W2.

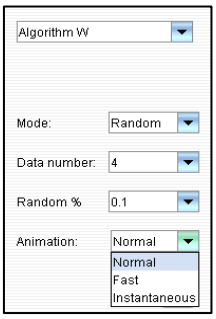
Και σε αυτό τον αλγόριθμο έγινε χρήση των κλάσεων Processor, MemorySlot και FloatingText. Σ' αυτή την περίπτωση όμως, οι κύκλοι που δημιουργεί η κλάση Processor χρησιμοποιούνται για το σχηματισμό των δυαδικών δέντρων του αλγορίθμου και συμβολίζουν τους κόμβους των δέντρων. Τα ορθογώνια που μπορούν να δημιουργηθούν με την κλάση MemorySlot χρησιμοποιήθηκαν για τη δημιουργία πίνακα, όπου κάθε κυψελίδα-θέση του πίνακα αντιπροσωπεύει έναν επεξεργαστή του αλγορίθμου και χρησιμοποιείται για να μας δείχνει την κατάσταση των επεξεργαστών, εάν δηλαδή είναι ενεργοί ή έχουν καταρρεύσει κατά τη διάρκεια εκτέλεσης του αλγορίθμου. Η κλάση FloatingText, χρησιμοποιήθηκε με τον ίδιο τρόπο, όπως στους προηγούμενους αλγορίθμους, για τη δημιουργία συμβολοσειρών που θα κινούνται με βάση τους άξονες χ και ψ για τη δημιουργία γραφικής αναπαράστασης με κίνηση, ώστε να δείχνει την κίνηση των επεξεργαστών καθώς διασχίζουν τα δέντρα.

Ο χρήστης έχει την επιλογή να τρέξει τον αλγόριθμο σε λειτουργία custom ή random. Δηλαδή μπορεί να επιλέξει το custom, εάν θέλει να είναι ο ίδιος υπεύθυνος για την επιλογή των επεξεργαστών που καταρρέουν, φτιάχνοντας τα δικά του σενάρια ή να επιλέξει το random, όπου, με βάση μία πιθανότητα, ο αλγόριθμος θα διαλέγει με τυχαίο τρόπο ποιοι επεξεργαστές θα καταρρεύσουν σε κάθε βήμα. Ένας επεξεργαστής μπορεί να καταρρεύσει είτε πριν να προλάβει να γράψει την τιμή 1 στον κόμβο που βρίσκεται είτε αμέσως μετά που θα τη γράψει.

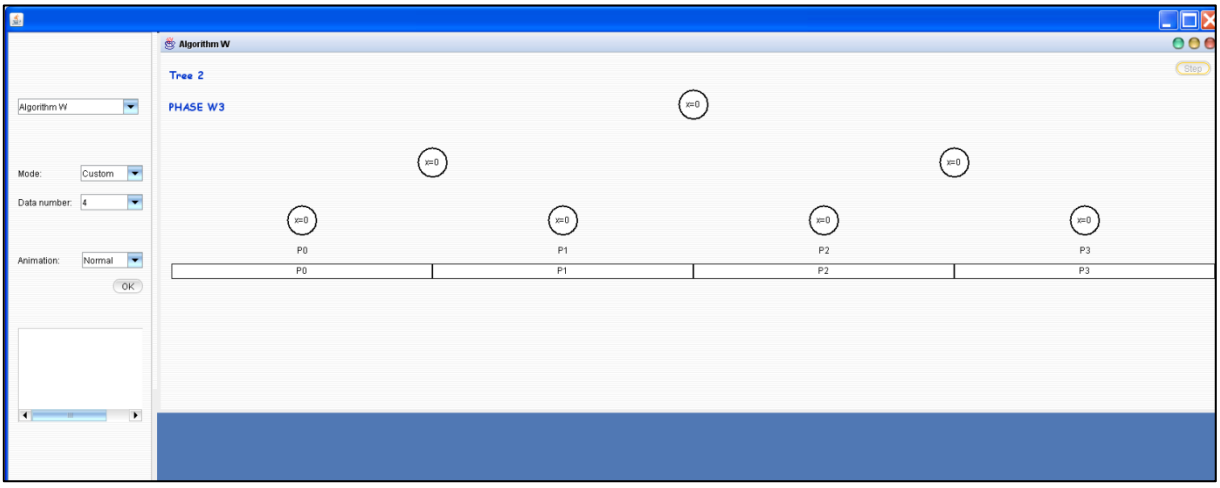
Εάν έχει επιλέξει την επιλογή custom, τότε μπορεί να σκοτώσει κάποιον επεξεργαστή κάνοντας κλικ πάνω στην κυψελίδα που τον αντιπροσωπεύει στον πίνακα των επεξεργαστών. Η υλοποίηση αυτής της λειτουργίας γίνεται με τον ίδιο τρόπο που υλοποιήθηκε στον αλγόριθμο παράλληλης άθροισης η λειτουργία να μπορεί ο χρήστης να αλλάζει τις τιμές στον πίνακα άθροισης με τη χρήση event listener (Υποκεφάλαιο 4.3.2). Εάν έχει επιλέξει την επιλογή random, τότε εμφανίζεται ένα καινούργιο Combo Box δίνοντας στον χρήστη την δυνατότητα να επιλέξει την πιθανότητα κάθε επεξεργαστή για να καταρρεύσει.(Σχήμα 5.1)

Επιπλέον λόγω του ότι ο αλγόριθμος αυτός έχει πολλά βήματα και φάσεις και ανάλογα με το σενάριο που μπορεί να τρέξει μπορεί να χρειαστεί να επαναλάβει αρκετές φορές τον βρόγχο με τις τέσσερις φάσεις, δώσαμε την δυνατότητα στον χρήστη, να επιλέξει την ταχύτητα με την οποία θα γίνεται η κίνηση στην γραφική αναπαράσταση. (Σχήμα 5.1)

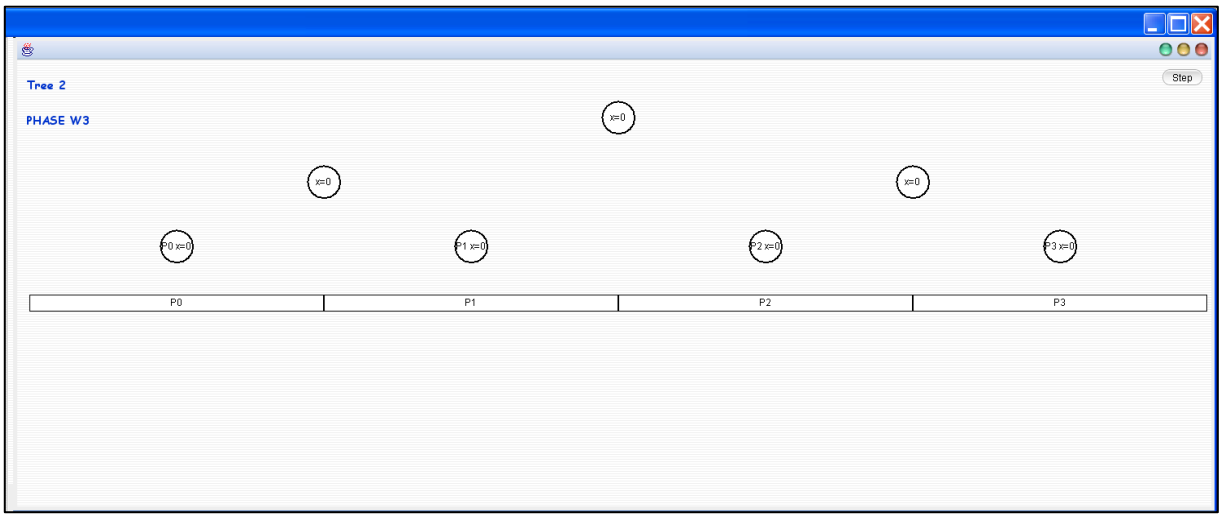
Στη φάση W3, γίνεται χρήση του διανύσματος δ_2, δηλαδή του δέντρου προόδου. Στη φάση αυτή, οι επεξεργαστές βρίσκονται στα φύλλα του δέντρου προόδου και αλλάζουν την τιμή των φύλλων από 0 σε 1. Τα σχήματα που ακολουθούν, δείχνουν τη γραφική αναπαράσταση της φάσης W3, την πρώτη φορά που εκτελείται, πριν ο αλγόριθμος εισέλθει στο βρόγχο. Η εφαρμογή μας μπορεί να τρέξει γραφική αναπαράσταση του αλγορίθμου W με n μέχρι 64, όπου n είναι τα φύλλα του δέντρου. Τα παραδείγματα στα σχήματα που ακολουθούν είναι με n=4, για να μπορέσουν να αναλυθούν οι φάσεις του αλγορίθμου με λιγότερα βήματα και να είναι πιο κατανοητή και πιο σύντομη η επεξήγηση.



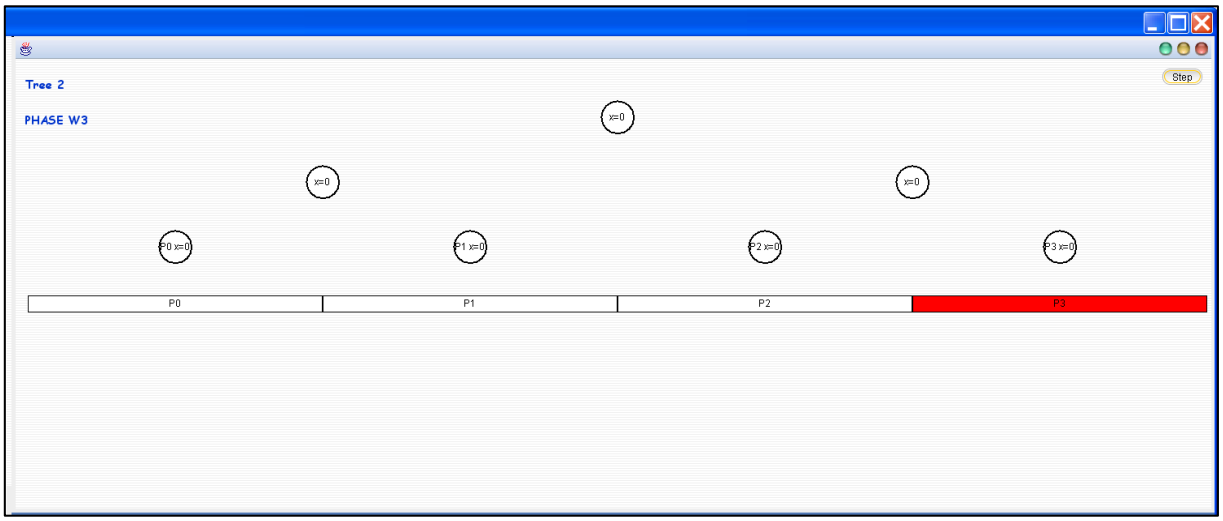
Σχήμα 5.1 – Αλγόριθμος W -Επιλογές για τον Χρήστη



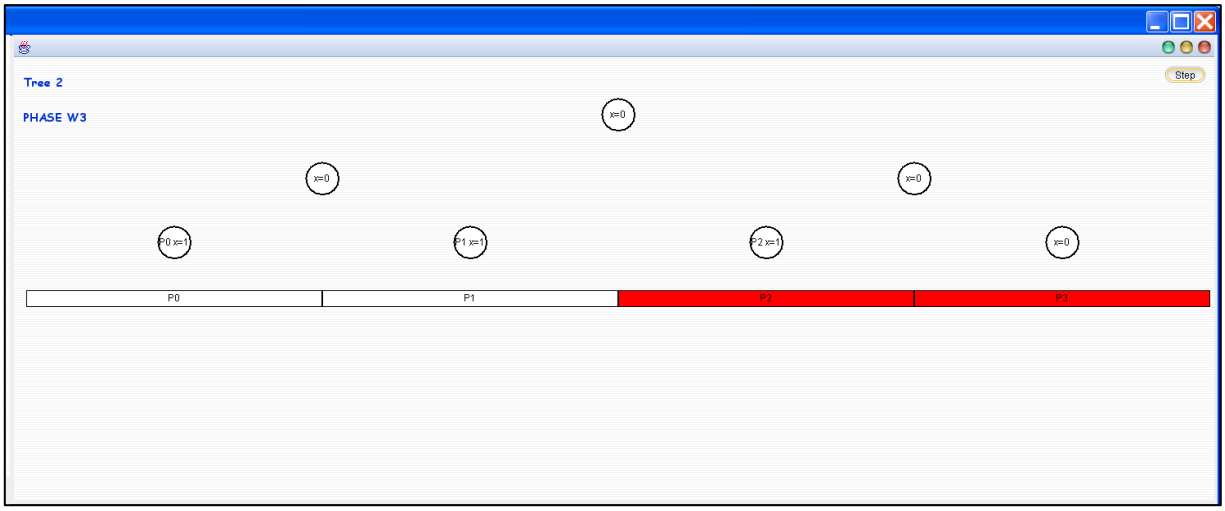
Σχήμα 5.2 – Αλγόριθμος W
Αρχική Ανάθεση Επεξεργαστών στα φύλλα του Δέντρου βάση του προσωπικού τους id



Σχήμα 5.3 – Φάση W3 - Αλγόριθμος W
Οι επεξεργαστές βρίσκονται στα φύλλα του δέντρου προόδου



Σχήμα 5.4 – Φάση W3 - Αλγόριθμος W
Ο επεξεργαστής P3 καταρρέει πριν να γράψει την τιμή 1



Σχήμα 5.5 – Φάση W3 - Αλγόριθμος W
Οι επεξεργαστές γράφουν την τιμή 1 στα φύλλα και ο επεξεργαστής P2 καταρρέει

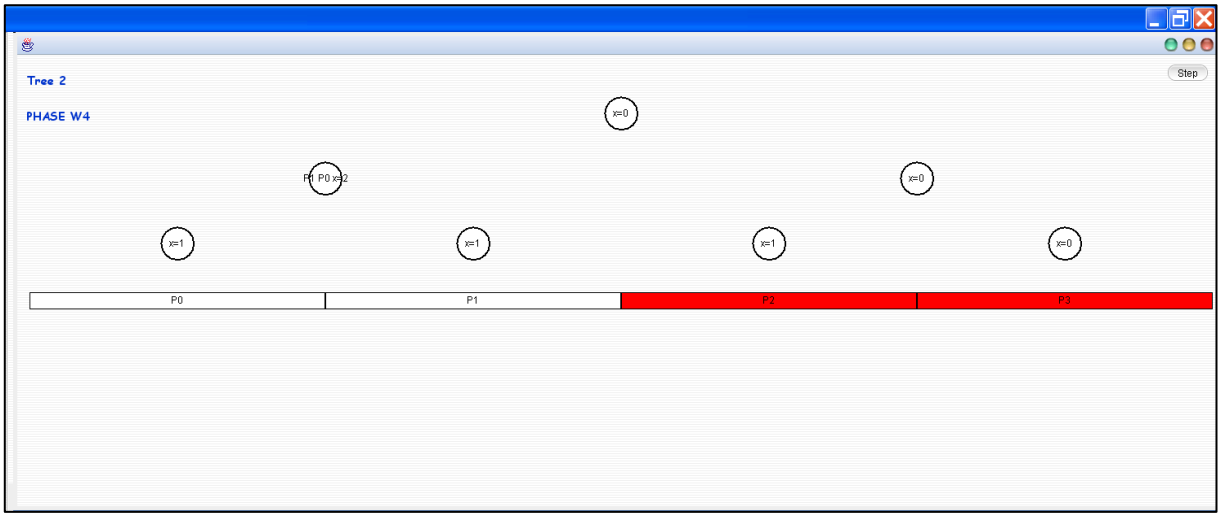
Όπως αναφέρθηκε προηγουμένως, στα πιο πάνω σχήματα (Σχήμα 5.2, 5.3, 5.4 και 5.5) φαίνεται η αναπαράσταση της φάσης W3 στην αρχή του αλγορίθμου. Σύμφωνα, λοιπόν, με τα πιο πάνω παραδείγματα, η ανάθεση των επεξεργαστών γίνεται με βάση το προσωπικό τους αριθμό (ο επεξεργαστής P0 στο πρώτο φύλλο κ.ο.κ.). Ο επεξεργαστής P3 καταρρέει πριν προλάβει να γράψει την τιμή 1 στο φύλλο που βρίσκεται, ενώ ο επεξεργαστής P2 καταρρέει αμέσως μετά που γράφει την τιμή 1 στο φύλλο του.

Η κίνηση που βλέπουμε στο Σχήμα 5.2 υλοποιήθηκε με τον ίδιο τρόπο που περιγράφηκε στο Υποκεφάλαιο 4.3.2.

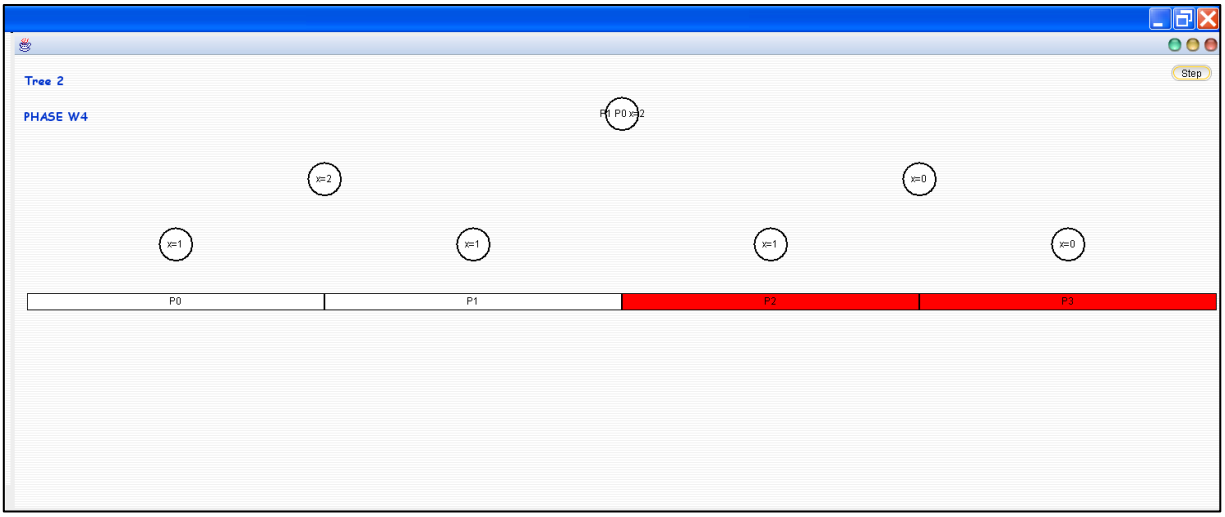
Στη φάση W4, που ακολουθεί τη φάση W3, γίνεται και πάλι χρήση του διανύσματος δ_2. Στη φάση αυτή οι επεξεργαστές, που παραμένουν ενεργοί, συνεχίζουν από εκεί που σταμάτησε η φάση W3, δηλαδή από τα φύλλα του δέντρου προόδου, αφού άλλαξε η τιμή τους σε 1 (σε όσα φύλλα οι επεξεργαστές δεν είχαν καταρρεύσει ακόμα), και ξεκινούν να εκτελούν μια παραλλαγή του μη βέλτιστου αλγορίθμου παράλληλης άθροισης μέχρι να φθάσουν στη ρίζα. Συγκεκριμένα, οι ενεργοί επεξεργαστές κινούνται προς τα πάνω, στον «πατρικό» κόμβο του κόμβου όπου βρίσκονται και στη συνέχεια υπολογίζουν το άθροισμα των «παιδιών» του κόμβου όπου βρίσκονται εκείνη τη στιγμή. Εάν ο αριθμός στη ρίζα είναι ίσως με το n, τότε ο αλγόριθμος τελειώνει σε αυτό το σημείο.

Τα σχήματα που ακολουθούν δείχνουν τη γραφική αναπαράσταση της φάσης W4, συνεχίζοντας, με βάση το προηγούμενο παράδειγμα, στα Σχήματα 5.2, 5.3, 5.4 και 5.5.

Σχήμα 5.6 – Φάση W4 - Αλγόριθμος W
Οι ενεργοί επεξεργαστές ανεβαίνουν ένα επίπεδο προς τα πάνω



Σχήμα 5.7 – Φάση W4 - Αλγόριθμος W
Οι ενεργοί επεξεργαστές υπολογίζουν το άθροισμα



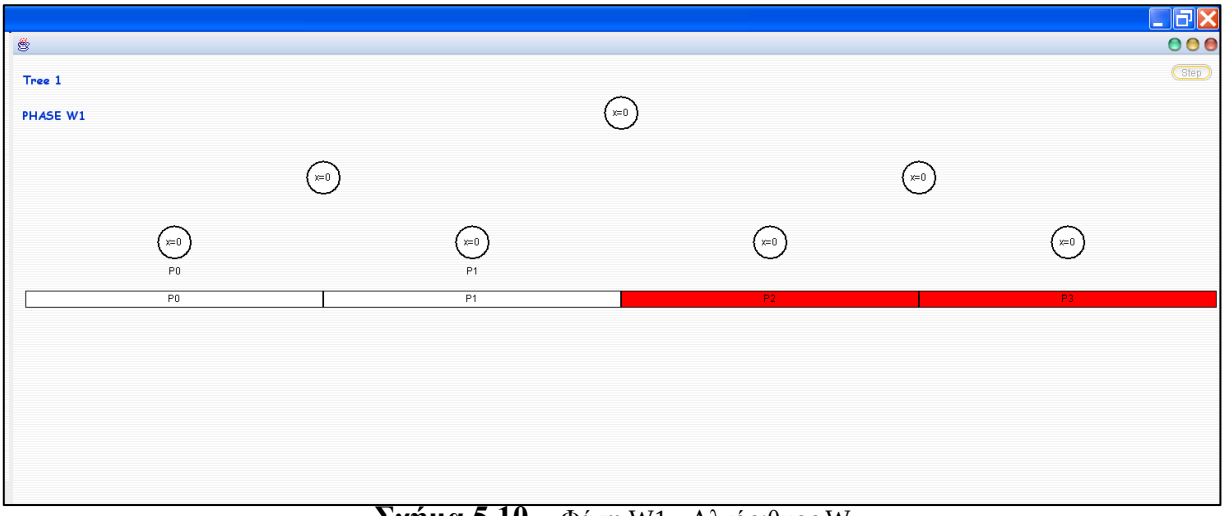
Σχήμα 5.8 – Φάση W4 - Αλγόριθμος W
Οι ενεργοί επεξεργαστές υπολογίζουν το άθροισμα στην ρίζα

Στο Σχήμα 5.8 έχει ολοκληρωθεί η φάση W4 και παραμένουν ενεργοί οι επεξεργαστές P0 και P1. Η τιμή στη ρίζα δεν είναι ίση με το n του αλγορίθμου, όπου n=4, αλλά είναι ίση με 2. Συνεπώς, ο αλγόριθμος δε σταματά εδώ, αλλά συνεχίζει και θα εισέλθει στο βρόγχο όπου θα πρέπει να εκτελέσει τις φάσεις W1, W2, W3 και W4 με τη σειρά. Επίσης, με βάση το Σχήμα 5.8, γίνεται η εξής παρατήρηση: η τιμή στη ρίζα του δέντρου προόδου, η οποία αντιπροσωπεύει τον αριθμό των φύλλων του δέντρου που έχουν την τιμή 1, είναι μία υποεκτίμηση σε σχέση με τον πραγματικό αριθμό των φύλλων που έχουν την τιμή 1.

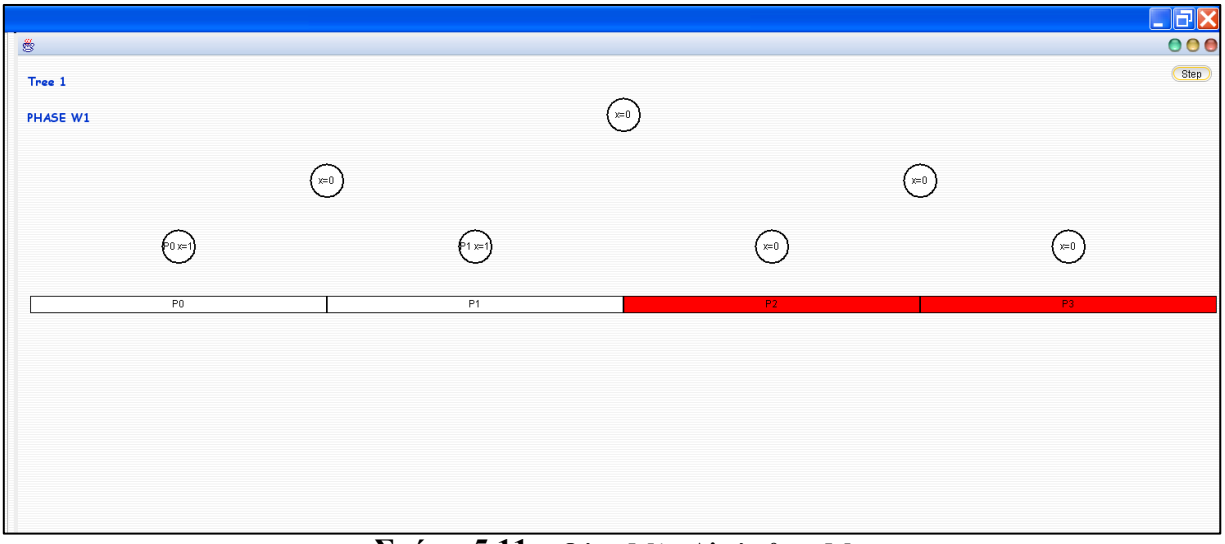
Αφού ο αλγόριθμος δεν τερματίστηκε στη φάση W4, τότε εισέρχεται στο βρόγχο και ξεκινάει η εκτέλεση της φάσης W1. Αυτή τη φορά γίνεται χρήση του διανύσματος δ_1, δηλαδή του δέντρου καταμέτρησης. Στη φάση αυτή οι επεξεργαστές που δεν έχουν καταρρεύσει τοποθετούνται στα φύλλα του δέντρου καταμέτρησης με βάση τον προσωπικό τους αριθμό. Στη συνέχεια, οι επεξεργαστές γράφουν την τιμή 1 στα φύλλα και εκτελούν μια παραλλαγή του μη βέλτιστου αλγορίθμου παράλληλης άθροισης μέχρι να φθάσουν στη ρίζα. Όπως και στη φάση W4, οι ενεργοί επεξεργαστές κινούνται προς τα πάνω, στον «πατρικό» κόμβο, του κόμβου όπου βρίσκονται, και στη συνέχεια υπολογίζουν το άθροισμα των «παιδιών» του κόμβου όπου βρίσκονται εκείνη τη στιγμή.

Τα σχήματα που ακολουθούν δείχνουν τη γραφική αναπαράσταση της φάσης W1, συνεχίζοντας, όπως και πριν, με βάση το προηγούμενο παράδειγμα.

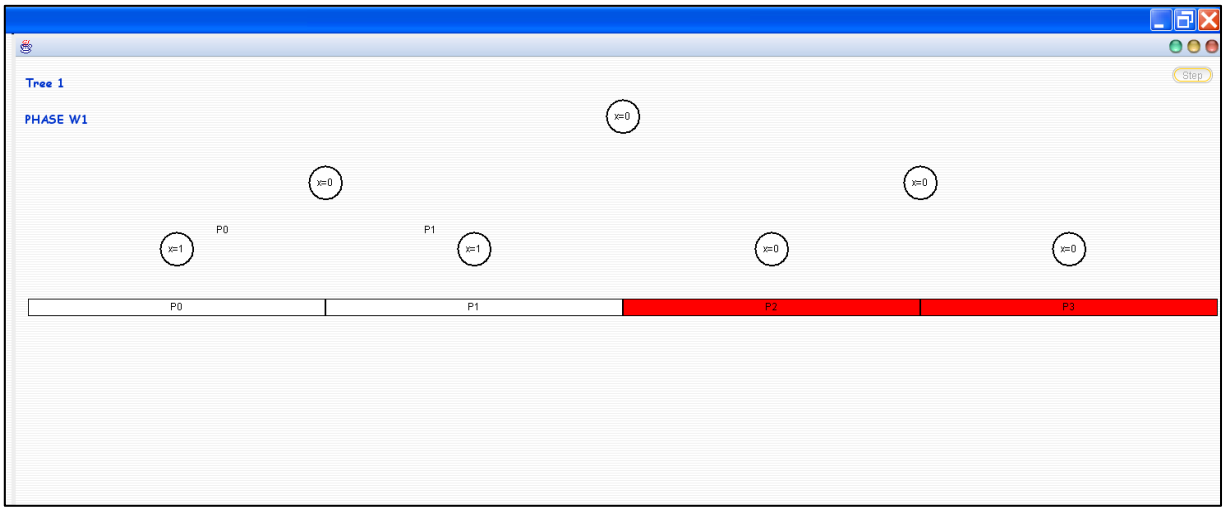
Σχήμα 5.9 – Φάση W1 - Αλγόριθμος W
Έγινε η Εναλλαγή Δέντρου (εμφάνιση δέντρου καταμέτρησης (δ_1))



Σχήμα 5.10 – Φάση W1 - Αλγόριθμος W
Ανάθεση Επεξεργαστών στα φύλλα του Δέντρου



Σχήμα 5.11 – Φάση W1 - Αλγόριθμος W
Οι επεξεργαστές γράφουν την τιμή 1 στα φύλλα



Σχήμα 5.12 – Φάση W1 - Αλγόριθμος W
Οι επεξεργαστές ανεβαίνουν ένα επίπεδο προς τα πάνω

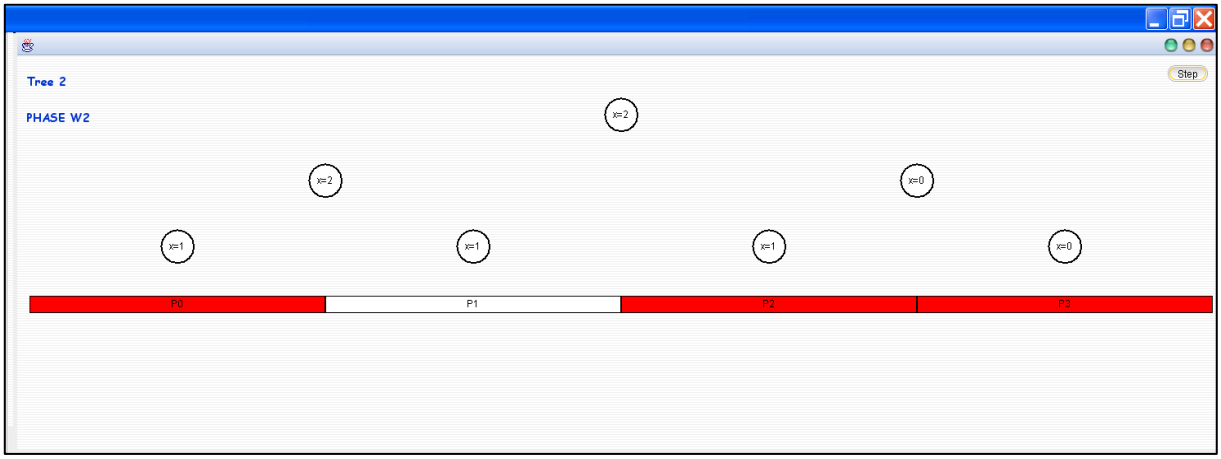
Σχήμα 5.13 – Φάση W1 - Αλγόριθμος W
Οι επεξεργαστές υπολογίζουν το άθροισμα και ο επεξεργαστής P0 καταρρέει

Σχήμα 5.14 – Φάση W1 - Αλγόριθμος W
Οι επεξεργαστές υπολογίζουν το άθροισμα της ρίζας

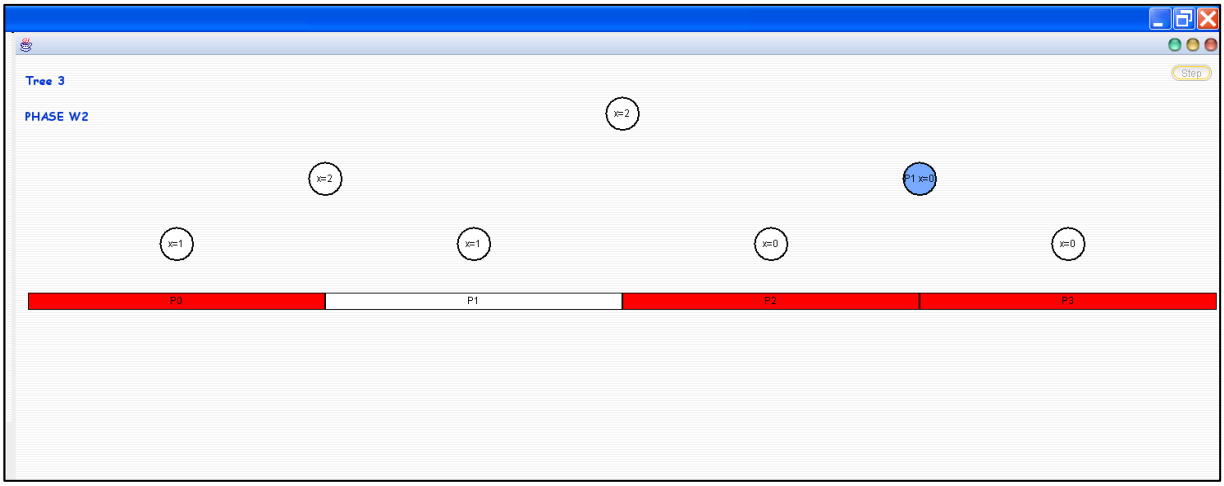
Στο Σχήμα 5.14 έχει ολοκληρωθεί η φάση W1 και παραμένει ενεργός μόνο ο επεξεργαστής P1, αφού ο επεξεργαστής P2 κατέρρευσε μετά τον υπολογισμό του αθροίσματος στον κόμβο 2, όπως φαίνεται στο σχήμα 5.13. Σύμφωνα με το σχήμα 5.14, η τιμή στη ρίζα είναι ίση με 2, γεγονός που μας παραπέμπει στην εξής παρατήρηση: η τιμή στη ρίζα του δέντρου καταμέτρησης, η οποία αντιπροσωπεύει τον αριθμό των επεξεργαστών που εξακολουθούν να είναι ενεργοί, είναι μία υποεκτίμηση σε σχέση με τον πραγματικό αριθμό των επεξεργαστών που δεν έχουν καταρρεύσει.

Μετά τη φάση W1 ακολουθεί η εκτέλεση της φάσης W2. Στη φάση W2 χρησιμοποιείται το δέντρο προόδου με τις τιμές που υπολόγισαν και αποθήκευσαν οι επεξεργαστές στη φάση W4, για να κάνει ανάθεση των ενεργών επεξεργαστών στα φύλλα του δέντρου. Για να μην αλλοιωθούν όμως οι τιμές στο δέντρο προόδου που αναπαριστά το διάνυσμα δ_2, χρησιμοποιείται ένα προσωρινό διάνυσμα δ_3, που αναπαρίσταται από ένα προσωρινό δέντρο, του οποίου οι τιμές υπολογίζονται με βάση το διάνυσμα δ2. Έτσι, οι επεξεργαστές ξεκινούν από τη ρίζα του προσωρινού αυτού δέντρου και κατεβαίνουν προς τα κάτω και διαχωρίζονται στα φύλλα του δέντρου, ανάλογα με τον αριθμό των υπολειπόμενων μηδενικών της λίστας, χρησιμοποιώντας τις τιμές του διανύσματος δ_3 και την εκτίμηση των ενεργών επεξεργαστών από το διάνυσμα δ_1.

Η γραφική αναπαράσταση της φάσης W2 φαίνεται στα σχήματα που ακολουθούν συνεχίζοντας, όπως και πριν, με βάση το προηγούμενο παράδειγμα.



Σχήμα 5.15 – Φάση W2 - Αλγόριθμος W
Εμφάνιση δέντρου προόδου και αντιγραφή του σε ένα προσωρινό δέντρο



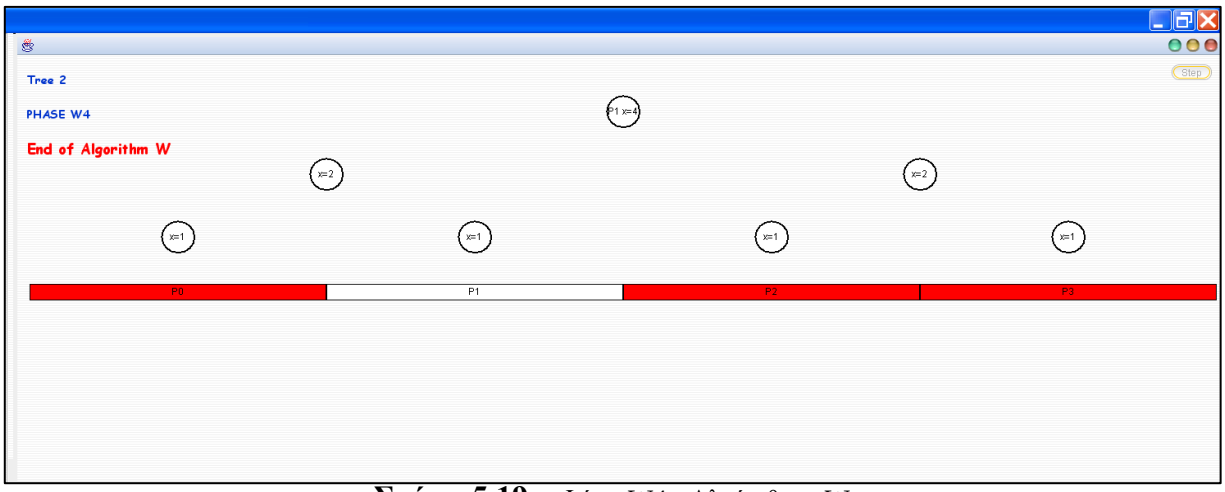
Σχήμα 5.16 – Φάση W2 - Αλγόριθμος W
Εμφάνιση προσωρινού δέντρου. Ο επεξεργαστής κατεβαίνει προς τα κάτω και επεξεργάζεται τις τιμές.

Σχήμα 5.17 – Φάση W2 - Αλγόριθμος W
Ανάθεση του επεξεργαστή P1 στο φύλλο του δέντρου (στον 7^{ο} κόμβο)

Στα πιο πάνω σχήματα παρακολουθήσαμε πώς λειτουργεί η φάση W2. Με το τέλος της φάσης W2, οι ενεργοί επεξεργαστές βρίσκονται και πάλι στα φύλλα του δέντρου προόδου. Στο παράδειγμά μας, λοιπόν, ο μοναδικός ενεργός επεξεργαστής, P1, βρίσκεται στα φύλλα προόδου και εκτελείται ξανά η φάση W3. Στη συνέχεια, εκτελείται η φάση W4 για να τελειώσει ένας κύκλος του βρόγχου. Εάν η τιμή στη ρίζα του δέντρου προόδου μετά τον τερματισμό της φάσης W4 είναι αυτή τη φορά ίση με το n του αλγορίθμου, τότε ο αλγόριθμος βγαίνει από το βρόγχο και τερματίζεται, αλλιώς συνεχίζει και εκτελεί ξανά το βρόγχο.

Ακολουθούν σχήματα τα οποία δείχνουν μόνο τα αποτελέσματα στο τέλος της κάθε φάσης, μέχρι ο αλγόριθμος να τερματιστεί και να τελειώσει το παράδειγμα, συνεχίζοντας με τη φάση W3.

Σχήμα 5.18 – Φάση W3 - Αλγόριθμος W
Ο επεξεργαστής P1 γράφει την τιμή 1 στο φύλλο που βρίσκεται



Σχήμα 5.19 – Φάση W4 - Αλγόριθμος W
Υπολογισμός αθροίσματος στην ρίζα – Τέλος Αλγορίθμου

Στα πιο πάνω σχήμα (Σχήμα 5.19) τελείωσε η φάση W4 του αλγορίθμου που είχε εκτελεστεί για δεύτερη φορά, έχοντας στη ρίζα την τιμή 4, που είναι ίση με το n του αλγορίθμου και έτσι ο αλγόριθμος τερματίστηκε.

5.3 Βέλτιστη Εκδοχή Αλγόριθμου W

5.3.1 Περιγραφή Αλγορίθμου

Για να έχουμε μία βέλτιστη εκδοχή του Αλγορίθμου W πρέπει να τροποποιήσουμε τον αλγόριθμο με τέτοιο τρόπο έτσι ώστε να έχουμε λιγότερο κόστος εκτέλεσης. Αυτό μπορούμε να το πετύχουμε με την μείωση του αριθμού των επεξεργαστών, το οποίο μπορεί επιτευχθεί κάνοντας κάποιες αλλαγές στην φάση W3 του Αλγόριθμου W. Πιο συγκεκριμένα, τα δέντρα που θα χρησιμοποιήσουμε αυτή την φορά θα έχουν n/logn φύλλα, αντί n, και κάθε φύλλο του δέντρου προόδου θα έχει logn στοιχεία. Επίσης αντί n επεξεργαστές θα έχουμε n/logn.

Συνεπώς, υλοποιώντας της αλλαγές αυτές για να εξακολουθεί να δουλεύει ορθά ο αλγόριθμος θα πρέπει να τον τροποποιήσουμε. Δηλαδή, οι επεξεργαστές θα εκτελούν σειριακά τις πράξεις στα logn στοιχεία των φύλλων που τους ανατίθενται. Επομένως στην φάση W3 θα γράψουν την τιμή 1 στα logn στοιχεία των φύλλων, σειριακά, και όταν πλέον θα έχουν γράψει την τιμή 1 σε όλα τα στοιχεία του φύλλου που τους αντιστοιχεί τότε θα αλλάξουν την τιμή του φύλλου που βρίσκονται από 0 σε 1.

Οι υπόλοιπες φάσεις (W1, W2, W4) θα παραμείνουν οι ίδιες. Δηλαδή θα χρησιμοποιούν τα δέντρα που αναφέραμε πιο πάνω, με n/logn φύλλα, αλλά δεν θα τους απασχολούν τα logn στοιχεία του κάθε φύλλου, το μόνο που χρειάζεται να γνωρίζουν είναι εάν η τιμή του φύλλου είναι 0 ή 1.

5.3.2 Γραφική Αναπαράσταση Αλγορίθμου

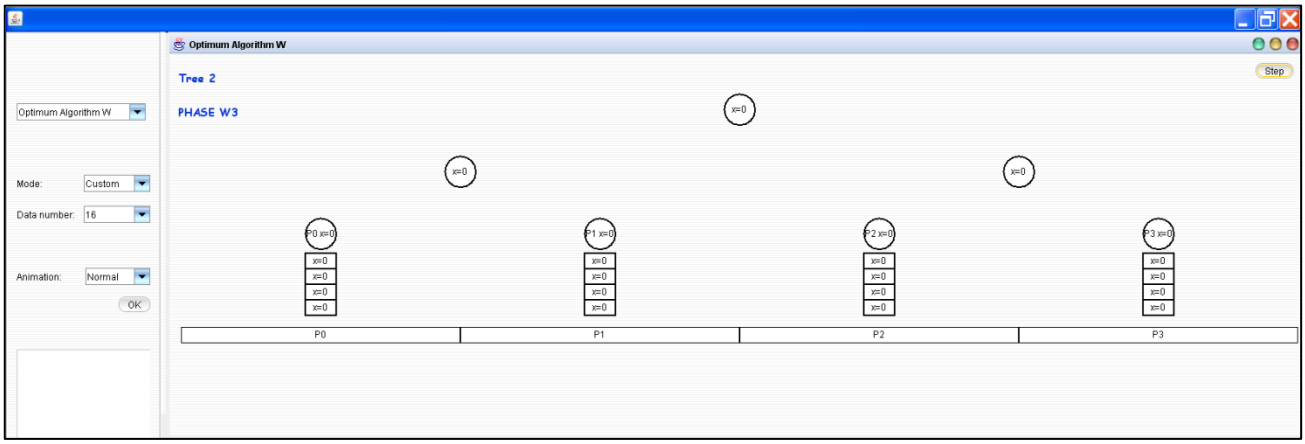
Μετά την υλοποίηση του αλγόριθμου W, εδώ παρουσιάζουμε την υλοποίηση της γραφικής αναπαράστασης της βέλτιστης εκδοχής του Αλγορίθμου W. Όπως και για τον αλγόριθμο W, χρησιμοποιήθηκε και πάλι «Internal Frame» για τη δημιουργία της φόρμας πάνω στην οποία θα τρέχει η γραφική αναπαράσταση του αλγορίθμου, για να μπορεί να εμφανίζεται στο «Desktop Pane» της αρχικής φόρμας της εφαρμογής. Από τα συστατικά που προσφέρει το

Java Swing χρησιμοποιήθηκε ένα «Button» (OK Button), το οποίο ονομάστηκε «step» και χρησιμοποιήθηκαν τρία «Labels» στα οποία εμφανίζονται διάφορα μηνύματα για να ενημερώνεται ο χρήστης.

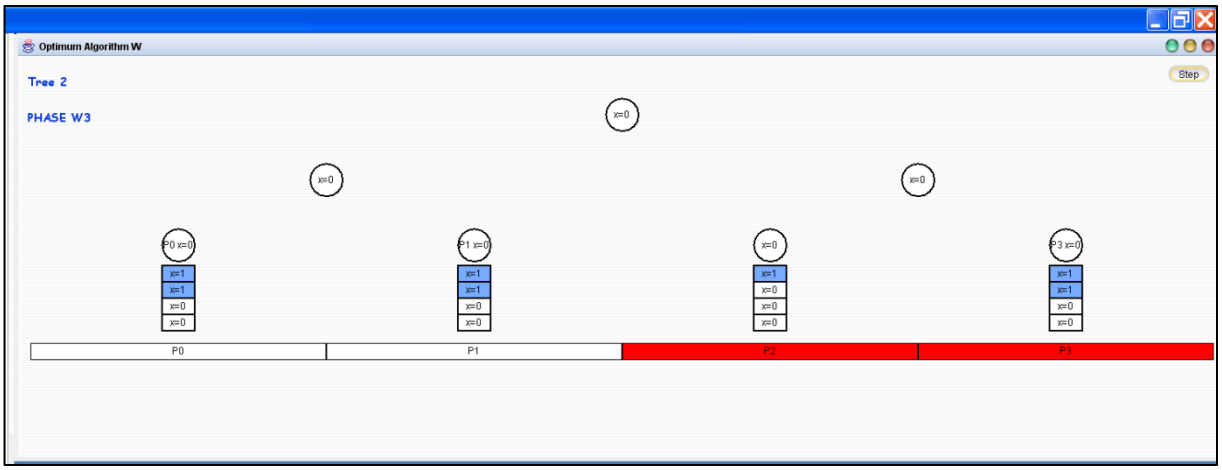
Σύμφωνα λοιπόν, με την περιγραφή της βέλτιστης εκδοχής του Αλγόριθμου W, η γραφική αναπαράσταση της δεν θα διαφέρει κατά πολύ από την γραφική αναπαράσταση του αλγόριθμου W. Εξακολουθούμε να έχουμε τις τέσσερις φάσεις όπως και πριν με μία διαφοροποίηση στην φάση W3, όπως αναφέραμε προηγουμένως, χρησιμοποιούμαι και πάλι δύο διανύσματα και δύο νοητά δέντρα, χρησιμοποιώντας τις κλάσεις «Processor», «MemorySlot» και «FloatingText». Με τη διαφορά ότι τώρα η κλάση «MemorySlot» δεν χρησιμοποιείται μόνο για την δημιουργία του πίνακα που διατηρεί την κατάσταση των επεξεργαστών αλλά και για την δημιουργία n/logn κάθετων πινάκων, ένα πίνακα για κάθε φύλλο, όπου κάθε θέση του πίνακα αντιπροσωπεύει ένα από τα logn στοιχεία του φύλλου.

Δίνεται και πάλι στον χρήστη η δυνατότητα να επιλέξει μεταξύ custom και random ακριβώς με τον ίδιο τρόπο που περιγράψαμε στο Υποκεφάλαιο 5.2.2, όπως και η δυνατότητα να επιλέξει την ταχύτητα της κίνησης στην γραφική αναπαράσταση.

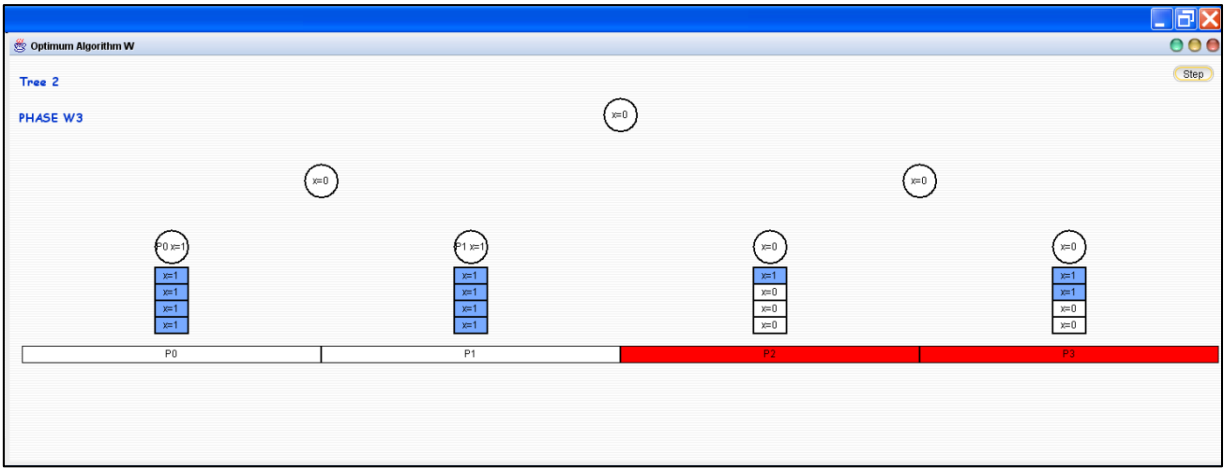
Θα ακολουθήσουν κάποια παραδείγματα από την γραφική αναπαράσταση της βέλτιστης εκδοχής του Αλγόριθμου W, στα οποία θα φαίνονται οι διαφορές σε σχέση με τον Αλγόριθμο W. Στα πιο κάτω παραδείγματα έχουμε n=16 και p= n/logn, άρα p=4 και σε κάθε φύλλο αντιστοιχούν 4 στοιχεία.



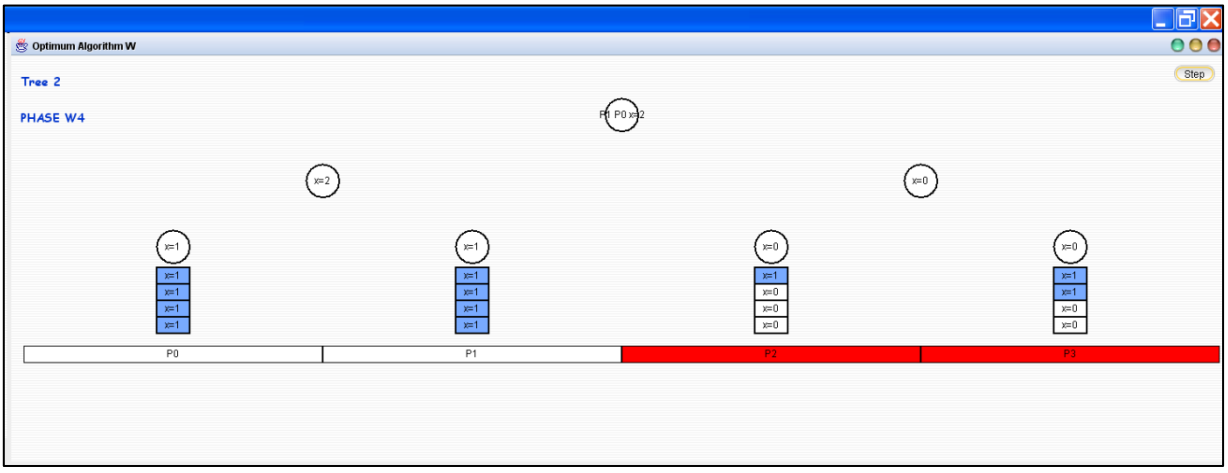
Σχήμα 5.20 – Φάση W3 – Βέλτιστη Εκδοχή Αλγορίθμου W
Οι επεξεργαστές βρίσκονται στα φύλλα του δέντρου προόδου



Σχήμα 5.21 – Φάση W3 – Βέλτιστη Εκδοχή Αλγορίθμου W
Ο επεξεργαστής P2 καταρρέει πριν γράψει την τιμή 1 ενώ ο P3 αμέσως μετά που γράφει την τιμή 1

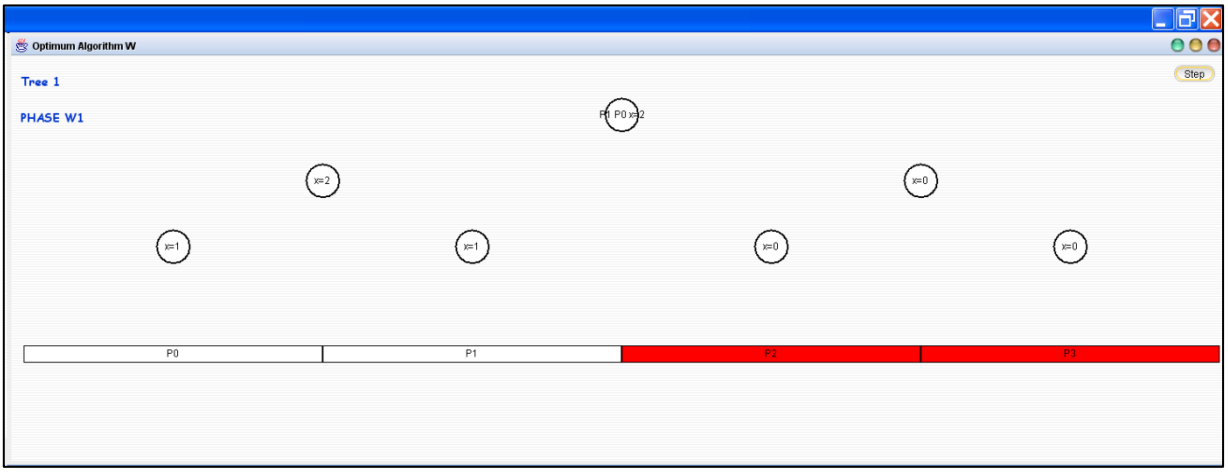


Σχήμα 5.22 – Φάση W3 – Βέλτιστη Εκδοχή Αλγορίθμου W
Οι επεξεργαστές P0, P1 έγραψαν την τιμή 1 σε όλα τα στοιχεία του φύλλου και άλλαξαν την τιμή του φύλλου σε 1

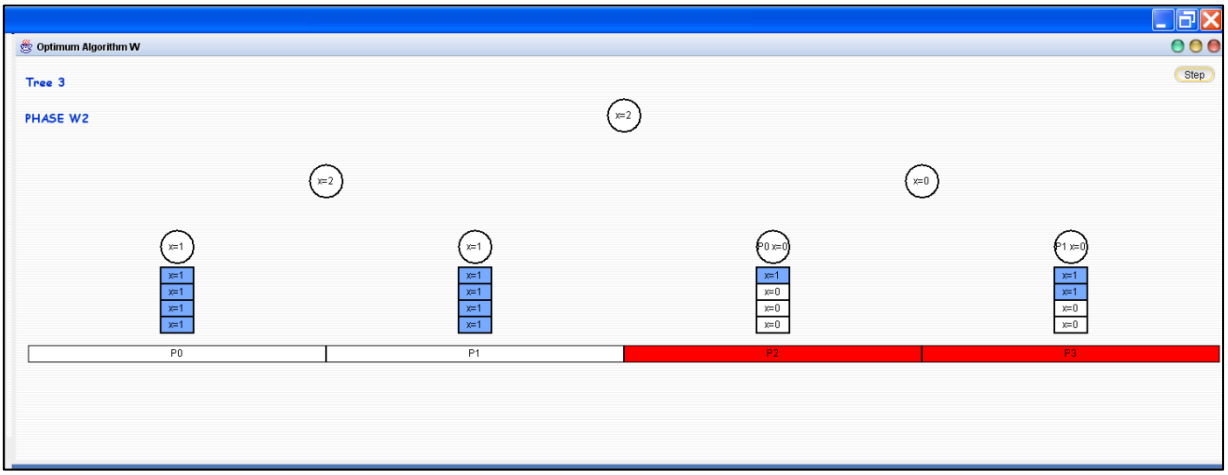


Σχήμα 5.23 – Φάση W4 – Βέλτιστη Εκδοχή Αλγορίθμου W
Οι ενεργοί επεξεργαστές υπολόγισαν το άθροισμα στην ρίζα

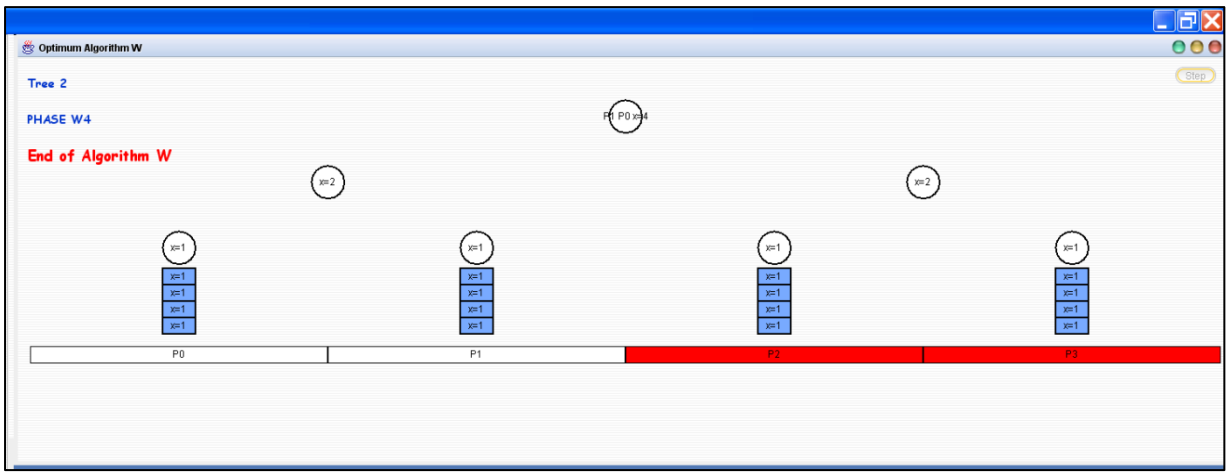
Όπως μπορούμε να παρατηρήσουμε στο Σχήμα 5.23, η φάση W4 δεν επηρεάστηκε καθόλου από την αλλαγή στην φάση W3. Έκανε κανονικά παράλληλη άθροιση των στοιχείων των κόμβων του δέντρου και αγνόησε τα logn στοιχεία των φύλλων, έλαβε υπόψη μόνο την τιμή στα φύλλα. Το ίδιο θα παρατηρήσουμε για τις φάσεις W1 και W2 στα επόμενα δύο παραδείγματα, Σχήματα 5.24 και 5.25. Στην φάση W1 αφού γίνετε χρήση του δέντρου καταμέτρησης δεν θα υπάρχουν στοιχεία στα φύλλα όπως προηγουμένως (Σχήμα 5.24).



Σχήμα 5.24 – Φάση W1 – Βέλτιστη Εκδοχή Αλγορίθμου W
Οι ενεργοί επεξεργαστές υπολόγισαν το άθροισμα στην ρίζα



Σχήμα 5.25 – Φάση W2 – Βέλτιστη Εκδοχή Αλγορίθμου W
Ισοζυγισμένη ανάθεση των επεξεργαστών P0, P1 στα φύλλα του δέντρου



Σχήμα 5.26 – Φάση W4 – Βέλτιστη Εκδοχή Αλγορίθμου W
Υπολογισμός αθροίσματος στην ρίζα – Τέλος Αλγορίθμου

Στο πιο πάνω σχήμα βλέπουμε πως είναι το τελικό αποτέλεσμα της βέλτιστης εκδοχής του αλγορίθμου W, όταν η εκτέλεση φθάσει στο τέλος. Αφού μετατράπηκαν όλα τα στοιχεία των φύλλων σε 1, που υπολοίπονταν, η τιμή των φύλλων έγινε 1 και το τελικό άθροισμα στην ρίζα του δέντρου προόδου 4 (ίσο με τον αριθμό των φύλλων του δέντρου).

Κεφάλαιο 6

Αλγόριθμος X

6.1 Πρόβλημα Write-All στο A-PRAM με n επεξεργαστές

Στο παρόν υποκεφάλαιο θα μελετηθεί το πρόβλημα Write-All [6] και στο μοντέλο A-PRAM [2,6] που, όπως έχει αναφερθεί στο Υποκεφάλαιο 2.4, είναι ένα μοντέλο όπου οι επεξεργαστές ενεργούν εντελώς ασυγχρόνιστα και μπορούν να γράψουν και να διαβάσουν σε οποιαδήποτε κυψελίδα της κοινόχρηστης μνήμης, όμως κάθε φορά η πρόσβαση στη μνήμη μπορεί να πάρει διαφορετικό χρόνο. Το πρόβλημα Write-All αναφέρεται στην περίπτωση όπου, δεδομένης μίας λίστας n στοιχείων, όπου αρχικά όλα τα στοιχεία έχουν την τιμή 0, ζητείται να μετατραπούν οι τιμές των στοιχείων σε 1 χρησιμοποιώντας p ασυγχρόνιστους επεξεργαστές στο μοντέλο A-PRAM.

6.2 Αλγόριθμος X

6.2.1 Περιγραφή Αλγορίθμου

Στον Αλγόριθμο X [2,6] επιλύεται το πρόβλημα Write-All με n στοιχεία, με τη χρήση n επεξεργαστών, οι οποίοι εργάζονται εντελώς ασύγχρονα. Στον αλγόριθμο αυτό χρησιμοποιείται ένα διάνυσμα μεγέθους 2n-1, το οποίο για την καλύτερη περιγραφή του αλγορίθμου θα το παρουσιάσουμε, αναπαριστώντας το ως ένα νοητό δυαδικό δέντρο με n φύλλα. Το διάνυσμα είναι αποθηκευμένο στις πρώτες 2n-1 κυψελίδες της κοινόχρηστης

μνήμης και έχουν αρχικά τιμή 0. Τα στοιχεία του προβλήματος Write-All είναι αποθηκευμένα στα φύλλα του δέντρου, δηλαδή στις κυψελίδες WA[n],…,WA[2n-1], όπου WA[] το διάνυσμα.

Αρχικά, οι επεξεργαστές τοποθετούνται στα φύλλα του δέντρου με την εξής διάταξη: ο επεξεργαστής P_i, όπου $1 \leq i \leq n$, τοποθετείται στην WA[n+i-1]. Έπειτα, ο αλγόριθμος έχει ως εξής: ο κάθε επεξεργαστής ανάλογα με την ταχύτητά του θα γράψει την τιμή 1 στο φύλλο που βρίσκεται.

Στην συνέχεια, θα ανεβεί ένα επίπεδο προς τα πάνω, δηλαδή στον πατέρα του φύλλου όπου βρισκόταν προηγουμένως. Διαβάζει την τιμή στον κόμβο όπου βρίσκεται τώρα (στον πατέρα) και, εάν είναι 1, τότε ανεβαίνει ξανά προς τα πάνω και καταλήγει στον πατέρα του. Εάν όμως η τιμή που θα διαβάσει είναι 0, τότε ελέγχει τις τιμές των κόμβων «παιδιών» του (ένα επίπεδο προς τα κάτω), δεξιά και αριστερά. Εάν και οι δύο κόμβοι έχουν την τιμή 1, τότε γράφει εκεί που βρίσκεται την τιμή 1 και προχωράει προς τα πάνω, καταλήγοντας στον πατέρα του. Εάν όμως ένας από τους δύο κόμβους έχει την τιμή 0, τότε ο επεξεργαστής κατεβαίνει ένα επίπεδο κάτω και μετατρέπει την τιμή του σε 1. Στη συνέχεια, ανεβαίνει ξανά ένα επίπεδο πάνω στον πατέρα, μετατρέπει και αυτού την τιμή σε 1 και ανεβαίνει ακόμα ένα επίπεδο πάνω.

Στην περίπτωση όμως που και τα δύο παιδιά του κόμβου όπου βρισκόμαστε έχουν την τιμή 0, τότε ο επεξεργαστής που βρίσκεται στον κόμβο θα επιλέξει σε ποιο από τα δύο παιδιά να πάει (δεξιά η αριστερά) με τον εξής υπολογισμό: ο επεξεργαστής θα υπολογίσει το δυαδικό του αριθμό βάση του προσωπικού του αριθμού και στη συνέχεια, με βάση το επίπεδο όπου βρίσκεται θα εξετάσει το αντίστοιχο bit αν είναι 0 ή 1. Εάν η τιμή του συγκεκριμένου bit είναι 1, τότε θα κατέβει ένα επίπεδο και θα πάει στο δεξί παιδί (2*where), εάν όμως είναι 0 θα πάει στο αριστερό παιδί (2*where-1).

Αυτή η διαδικασία, με τους επεξεργαστές να κινούνται πάνω-κάτω στο δέντρο και να γράφουν την τιμή 1 θα συνεχιστεί μέχρις ότου έστω και ένας επεξεργαστής διαβάσει την τιμή 1 στην ρίζα του δέντρου (δηλαδή WA[1]=1).

Ακολουθεί ένα κομμάτι κώδικα όπου γίνεται η επίλυση του προβλήματος Write-All μεγέθους n, με τον Αλγόριθμο X, χρησιμοποιώντας n επεξεργαστές [2]:

```
Processors P1, ..., Pn do asynchronously
      Shared array WA[1…2n-1]
      Pi: where = n + i - 1
           while WA[1] ≠ 1 do
            if (WA[where] ≠ 1 && where > n-1) then  % σε φύλλο
                  WA[where]=1
                  where = ⌊where/2⌋  %ανεβαίνει επίπεδο
            else if (WA[where] ≠ 1 && where ≤ n-1) then  %όχι σε φύλλο
                  if (WA[2*where] = 1 && WA[2*where+1] = 1) then
                        WA[where]=1  %τελειωμένο υποδέντρο
                        where = ⌊where/2⌋ %ανεβαίνει επίπεδο
                  if (WA[2*where] = 1 && WA[2*where+1] = 0) then
                        where = 2*where+1  %κατεβαίνει επίπεδο
                  if (WA[2*where] = 0 && WA[2*where] = 1) then
                        where = 2*where  %κατεβαίνει επίπεδο
                  if (WA[2*where] = 0 && WA[2*where] = 0) then
                        if (i_log(where)=0) then
                              where = 2*where  %στο αριστερό παιδί
                        else where = 2*where + 1  %στο δεξί παιδί
            end if
      end while
end do
```

(το $i_{log(k)}$ συμβολίζει την τιμή του log(k)^{στου} διφίου της log(n)-δυαδικής αναπαράστασης του i)

6.2.2 Γραφική Αναπαράσταση Αλγορίθμου

Μετά την υλοποίηση της γραφικής αναπαράστασης του αλγορίθμου W, εδώ παρουσιάζουμε την υλοποίηση του Αλγορίθμου X.

Όπως και για τους άλλους αλγόριθμους, χρησιμοποιήθηκε και πάλι «Internal Frame» για τη δημιουργία της φόρμας πάνω στην οποία θα τρέχει η γραφική αναπαράσταση του Αλγορίθμου X, έτσι ώστε να μπορεί να εμφανίζεται στο «Desktop Pane» της αρχικής φόρμας της εφαρμογής. Πάνω στη φόρμα τοποθετήθηκε ένα «Button» (OK Button), το οποίο ονομάστηκε «step» και θα είναι αυτό που θα συγχρονίζει την κίνηση στη γραφική αναπαράσταση και τα βήματα του αλγορίθμου. Επίσης χρησιμοποιήθηκε ένα «Label» στο οποίο εμφανίζονται διάφορα μηνύματα για να ενημερώνεται ο χρήστης για θέματα, όπως για παράδειγμα τον τερματισμό του αλγορίθμου.

Σύμφωνα με την περιγραφή που δόθηκε στο Υποκεφάλαιο 6.2.1, ο αλγόριθμος X επιλύει το πρόβλημα Write-All με n στοιχεία, με τη χρήση n επεξεργαστών, οι οποίοι εργάζονται εντελώς ασύγχρονα. Στον αλγόριθμο αυτό χρησιμοποιείται ένα διάνυσμα 2n-1, το οποίο αναπαρίσταται ως ένα νοητό πλήρες δυαδικό δέντρο με n φύλλα. Έτσι, για την υλοποίηση της γραφικής αναπαράστασης του αλγορίθμου αυτού, χρησιμοποιήθηκε ένας πίνακας για την αποθήκευση του διανύσματος και το δέντρο σχεδιάστηκε με την κλάση «Processor». Η κλάση αυτή, όπως αναφέρθηκε και στα προηγούμενα υποκεφάλαια, δημιουργεί κύκλους με τη χρήση των Graphics. Στην περίπτωση αυτή κάθε κύκλος που δημιουργείται αντιπροσωπεύει έναν κόμβο του δέντρου.

Επίσης, έγινε χρήση και των άλλων δύο κλάσεων που αναφέρθηκαν στα προηγούμενα υποκεφάλαια, «MemorySlot» και «FloatingText». Τα ορθογώνια που μπορούν να δημιουργηθούν με την κλάση «MemorySlot» χρησιμοποιήθηκαν για τη δημιουργία πίνακα, όπου κάθε κυψελίδα-θέση του πίνακα αντιπροσωπεύει έναν επεξεργαστή του αλγορίθμου. Η χρήση του πίνακα αυτού είναι μόνο για λόγους αισθητικής. Παρουσιάζεται μόνο στο πρώτο βήμα του αλγορίθμου όταν γίνεται η ανάθεση επεξεργαστών στα φύλλα του δέντρου και μετά φεύγει.

Η κλάση «FloatingText» χρησιμοποιήθηκε με τον ίδιο τρόπο όπως στους προηγούμενους αλγορίθμους. Δηλαδή, για τη δημιουργία συμβολοσειρών που θα κινούνται με βάση τους άξονες x και y για τη δημιουργία γραφικής αναπαράστασης με κίνηση, ώστε να δείχνει την κίνηση των επεξεργαστών καθώς διασχίζουν τα δέντρα.

Ο χρήστης έχει την επιλογή αν θέλει να επιλέξει έναν από τους επεξεργαστές να κινείται πιο γρήγορα από τους άλλους και η ταχύτητα των υπόλοιπων επεξεργαστών να υπολογιστεί με τυχαίο τρόπο από την εφαρμογή ή να επιλέξει τη λειτουργία random και να υπολογιστεί, αυτή την φορά, η ταχύτητα όλων των επεξεργαστών με τυχαίο τρόπο.

Επιπλέον λόγω του ότι και αυτός ο αλγόριθμος έχει πολλά βήματα, όπως και ο Αλγόριθμος W (Υποκεφάλαιο 5.2.2), και ανάλογα με το σενάριο που μπορεί να τρέξει μπορεί να χρειαστεί ένας επεξεργαστής να διανύσει όλους τους κόμβους του δέντρου μέχρι να φθάσει στο τέλος ο αλγόριθμος, δώσαμε την δυνατότητα στον χρήστη, να επιλέξει την ταχύτητα με την οποία θα γίνεται η κίνηση στην γραφική αναπαράσταση.

Όπως αναφέρθηκε προηγουμένως, οι επεξεργαστές τρέχουν ασύγχρονα. Άρα, τη στιγμή που ο ένας επεξεργαστής έχει γράψει την τιμή 1 στο φύλλο του και έχει προχωρήσει στο πιο πάνω επίπεδο, ένας άλλος επεξεργαστής μπορεί να βρίσκεται ακόμη στο φύλλο του και να μην έχει γράψει καν την τιμή 1. Για να υλοποιηθεί όμως αυτό γραφικά και να έχουμε γραφική αναπαράσταση του αλγορίθμου με κίνηση και να δείχνουμε κάθε φορά την κίνηση ενός επεξεργαστή όταν κινείται για να αλλάξει επίπεδο, πρέπει με κάποιο τρόπο να συγχρονίσουμε τον αλγόριθμο υλοποίησης, ενώ παράλληλα η γραφική αναπαράσταση να δίνει στο χρήστη την εικόνα ότι οι επεξεργαστές κινούνται ασύγχρονα και παράλληλα.

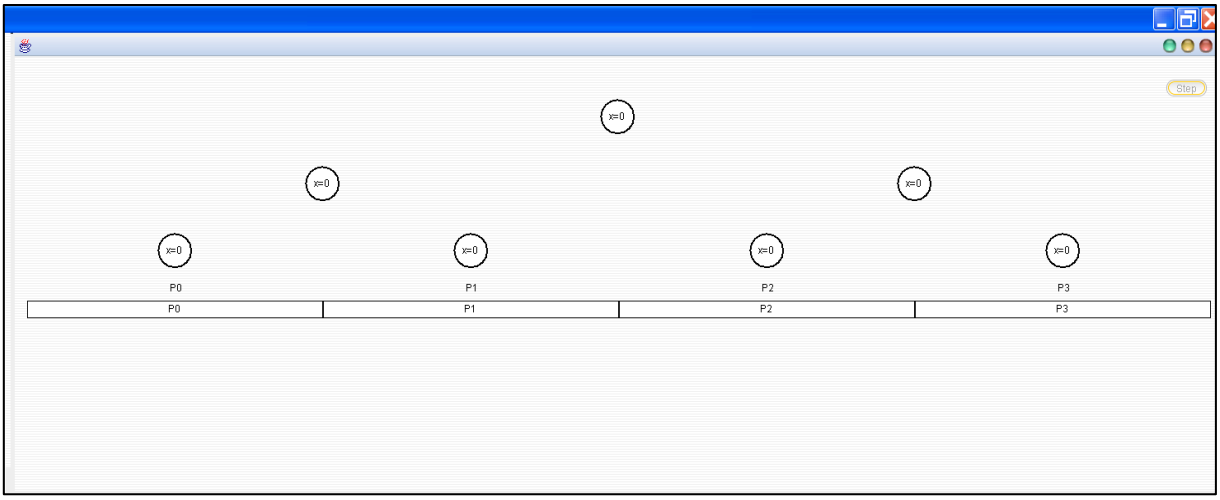
Για το λόγο αυτό, για τον κάθε επεξεργαστή υπάρχουν δύο πεδία στο πρώτο από τα οποία αποθηκεύεται η ταχύτητα του επεξεργαστή και στο δεύτερο το βήμα που εκτέλεσε κάποια πράξη για τελευταία φορά. Ένας επεξεργαστής, σε ένα βήμα του αλγορίθμου, μπορεί είτε να γράψει την τιμή 1 στον κόμβο που βρίσκεται είτε να κινηθεί σε άλλο κόμβο, όχι όμως και τα δύο. Η ταχύτητά του είναι μια τιμή από 1 μέχρι 100, η οποία επιλέγεται με τυχαίο τρόπο. Το δεύτερο πεδίο του επεξεργαστή, πριν ξεκινήσει η εκτέλεση του αλγορίθμου, αρχικοποιείται με την τιμή που έχει ο κάθε επεξεργαστής για ταχύτητα. Δηλαδή, στην αρχή τα δύο πεδία του κάθε επεξεργαστή είναι ίσα μεταξύ τους. Η μεταβλητή που αποθηκεύει τα βήματα, για λόγους ευκολίας ας την ονομάσουμε round, έχει αρχική τιμή 0, ενώ μία άλλη μεταβλητή, που ονομάσαμε minSpeed, αρχικά παίρνει την τιμή Integer.MAX_VALUE, δηλαδή το μεγαλύτερο ακέραιο αριθμό που υπάρχει.

Όταν ο χρήστης πατήσει το κουμπί step, τότε ο αλγόριθμος ελέγχει ποιος επεξεργαστής έχει τη μικρότερη τιμή στο δεύτερο πεδίο του, από όλους και από την τιμή minSpeed και τότε η μεταβλητή minSpeed γίνεται ίση με αυτή την τιμή. Στη συνέχεια, όποιοι επεξεργαστές έχουν στο δεύτερο πεδίο τους τιμή ίση με τη minSpeed, τότε μπορούν να εκτελέσουν την επόμενη τους πράξη. Στην τιμή του δεύτερου πεδίου των επεξεργαστών που εκτέλεσαν μία πράξη σε αυτό το γύρο, προσθέτεται η τιμή που έχουν στο πρώτο πεδίο τους, δηλαδή η ταχύτητά τους. Στην μεταβλητή round προστίθεται η τιμή που έχει η μεταβλητή minSpeed και η μεταβλητή minSpeed αρχικοποιείται και πάλι με την τιμή Integer.MAX_VALUE.

Το κομμάτι κώδικα που ακολουθεί είναι ένα μέρος από την υλοποίηση της γραφικής αναπαράστασης του αλγορίθμου Χ και πιο συγκεκριμένα είναι το κομμάτι που υλοποιεί ότι περιγράψαμε στις προηγούμενες δύο παραγράφους.

```
…
int round = 0;
int[][] cpuSpeed;
…
public AlgorithmX(…){
    …
    cpuSpeed = new int[4][CNum];
    int temp2 = totalNodes - CNum;
    for (int i = 0; i < CNum; i++) {
      cpuSpeed[0][i] = 0; //cpuSpeed
      cpuSpeed[1][i] = 0; //cpuLastRoundUsed
      cpuSpeed[2][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla
      tou dentrou)
      cpuSpeed[3][i] = 1; // if the cpu comes to an end the value becomes 0
      temp2++; }
    Random generator = new Random();
    for (int i = 0; i < CNum; i++) {
      int r = generator.nextInt(100) + 1;
      cpuSpeed[0][i] = r;
      cpuSpeed[1][i] = r;
    }
}
private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
    int minSpeed = Integer.MAX_VALUE;
    …
    for (int i = 0; i < CNum; i++) {
      if (cpuSpeed[1][i] < minSpeed && cpuSpeed[3][i] == 1) {
        minSpeed = cpuSpeed[1][i]; }
    }
    for (int i = 0; i < CNum; i++) {
      if (cpuSpeed[1][i] == minSpeed && cpuSpeed[3][i] == 1) {
        vCPUReady.add(i);
        runner = new Thread(this);
        runner.start();
        cpuSpeed[1][i] += cpuSpeed[0][i];
        end++; }
    }
    round += minSpeed;
    …
}
```

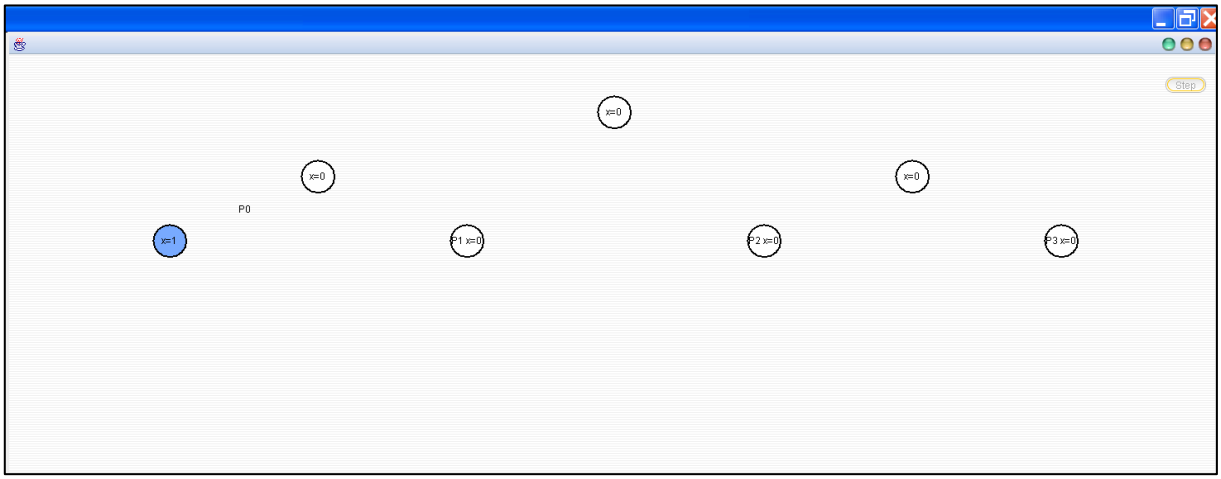
Μιας και οι επεξεργαστές σε αυτό τον αλγόριθμο κινούνται ασύγχρονα δεν υπάρχει κάποια συγκεκριμένη διαδικασία όπως στον αλγόριθμο W, που είχε τις τέσσερις φάσεις να περιγραφούν. Έτσι, θα τρέξουμε ένα παράδειγμα στην εφαρμογή και θα περιγράψουμε τις κινήσεις που γίνονται. Η εφαρμογή μας μπορεί να τρέξει γραφική αναπαράσταση του αλγορίθμου X με n μέχρι 64, όπου n είναι τα φύλλα του δέντρου. Το παράδειγμα όμως, στα σχήματα που ακολουθούν, θα έχει n=4, για να μπορέσουν να αναλυθούν όλα τα βήματα των επεξεργαστών και οι αποφάσεις που παίρνει ο αλγόριθμος για την κατεύθυνση των επεξεργαστών με βάση την περιγραφή του αλγορίθμου X, που έγινε στο Υποκεφάλαιο 6.2.1. Επίσης, θα επιλεγεί ο επεξεργαστής P0 να είναι πιο γρήγορος από τους υπόλοιπους, έτσι ώστε να μπορέσουμε να έχουμε στο παράδειγμα μας όλα τα πιθανά σενάρια-βήματα που αναφέρθηκαν στο Υποκεφάλαιο 6.2.1, για την επιλογή του επόμενου κόμβου που θα επισκεφθεί ο επεξεργαστής.



Σχήμα 6.1 – Αλγόριθμος X
Ανάθεση Επεξεργαστών στα φύλλα του Δέντρου βάση του προσωπικού τους αριθμού

Σχήμα 6.2 – Αλγόριθμος Χ

Ο επεξεργαστής P0 γράφει την τιμή ‘1’ στο αρχικό φύλλο που του ανάθεσαν. Οι υπόλοιποι επεξεργαστές ακόμα.

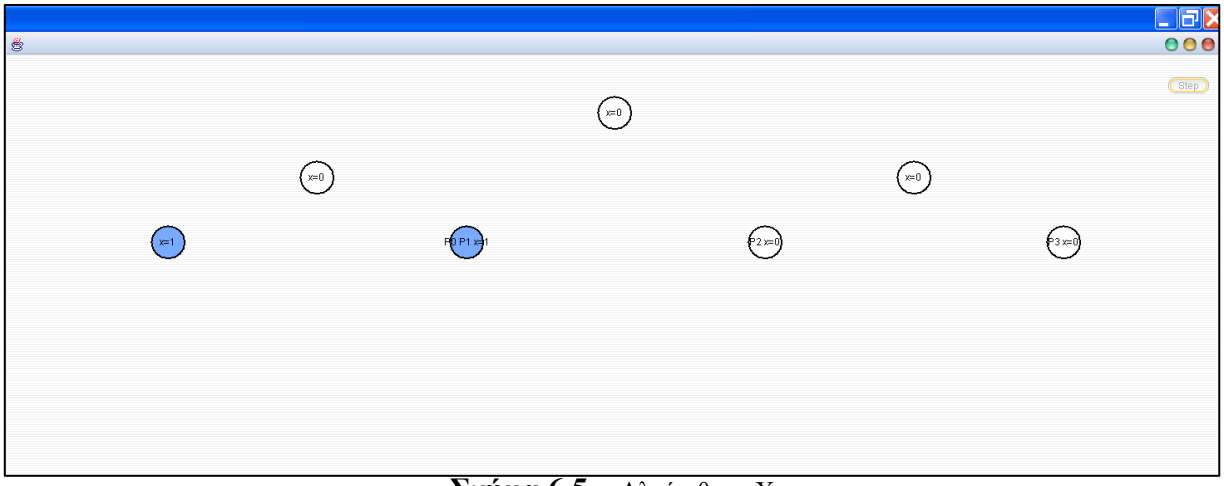


Σχήμα 6.3 – Αλγόριθμος Χ

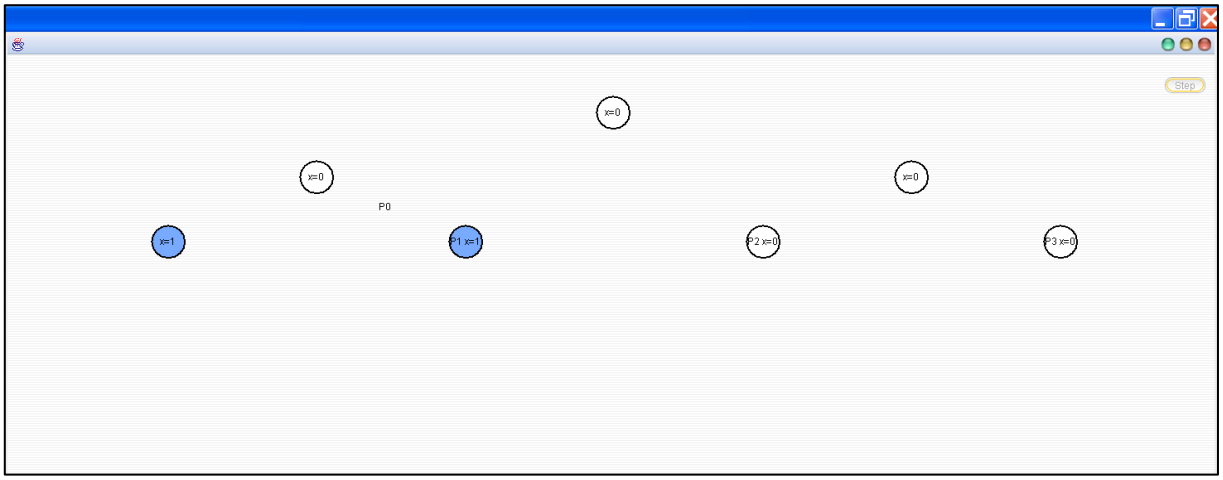
Ο P0 βρίσκεται σε φύλλο που έχει την τιμή ‘1’, έτσι κινείται 1 επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

Σχήμα 6.4 – Αλγόριθμος Χ

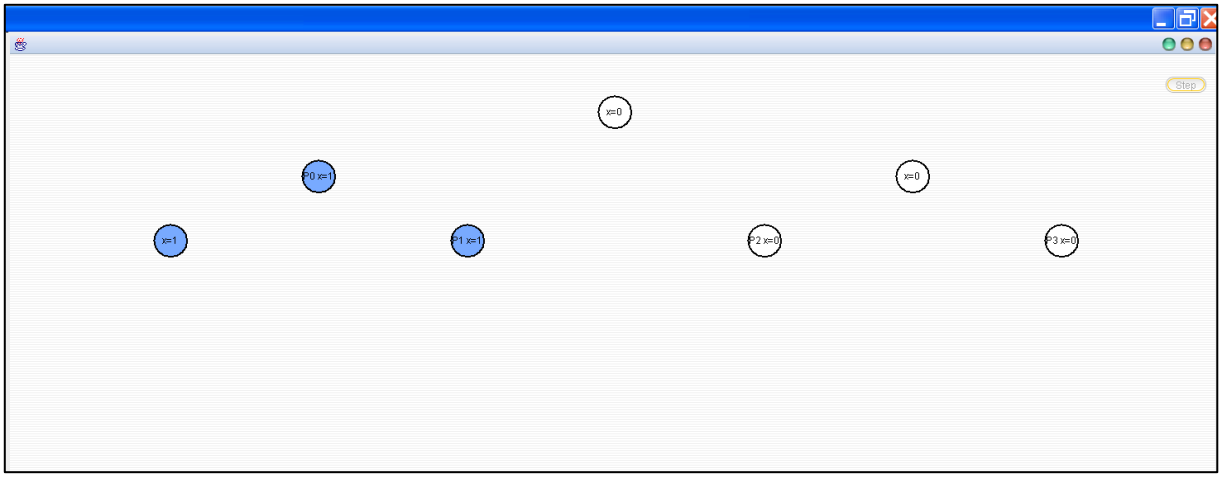
Ο P0 είναι σε κόμβο με τιμή ‘0’, το δεξί παιδί του είναι ‘0’, έτσι κατεβαίνει σε αυτό.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



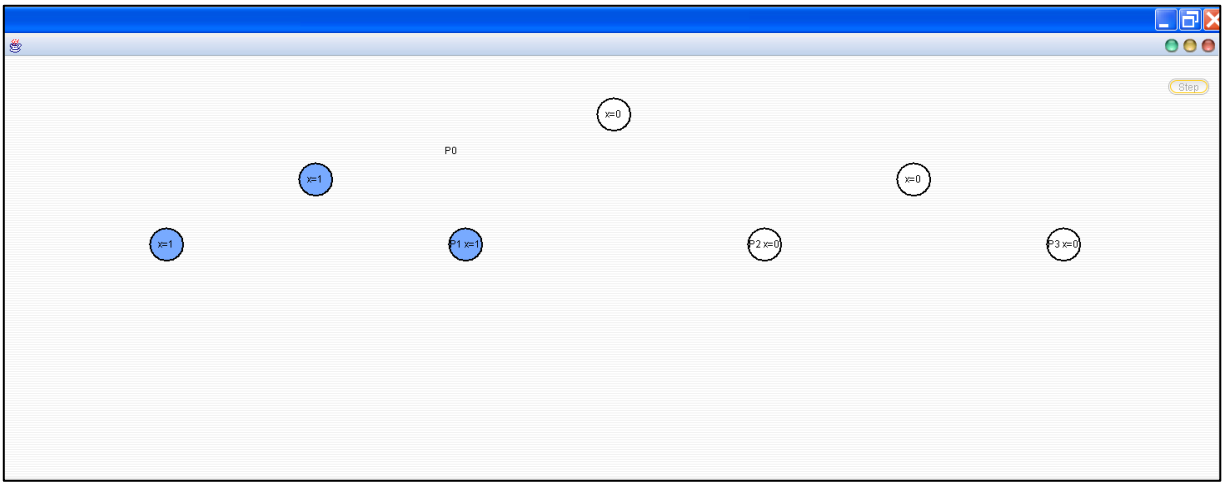
Σχήμα 6.5 – Αλγόριθμος X
Ο P0 βρίσκεται σε φύλλο που έχει την τιμή ‘0’, έτσι αλλάζει την τιμή σε ‘1’.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



Σχήμα 6.6 – Αλγόριθμος X
Ο P0 βρίσκεται σε φύλλο που έχει την τιμή ‘1’, έτσι κινείται 1 επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

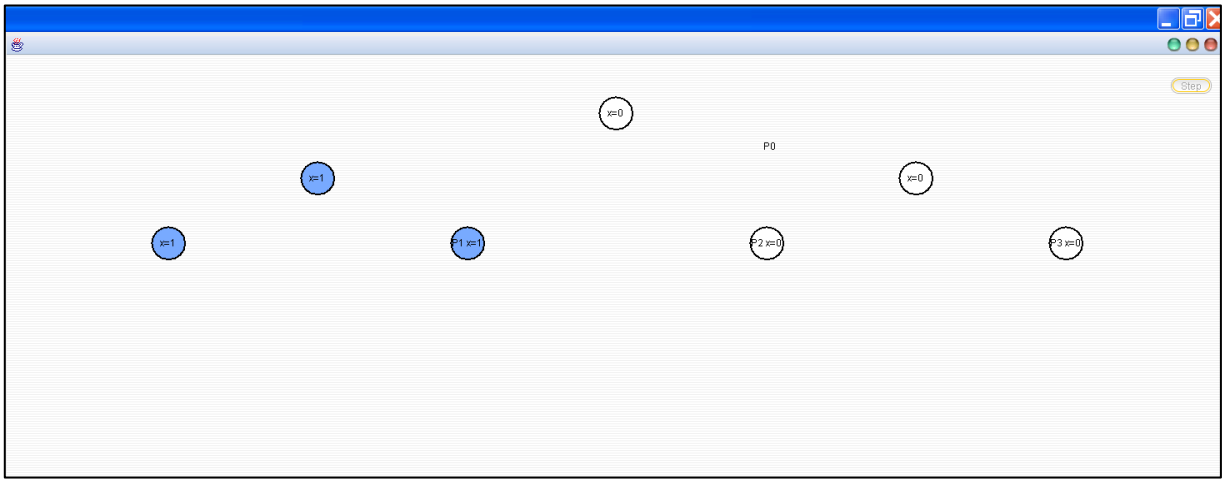


Σχήμα 6.7 – Αλγόριθμος X
Ο P0 βρίσκεται σε κόμβο με τιμή ‘0’, τα παιδιά του έχουν τιμή ‘1’, έτσι γράφει ‘1’.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



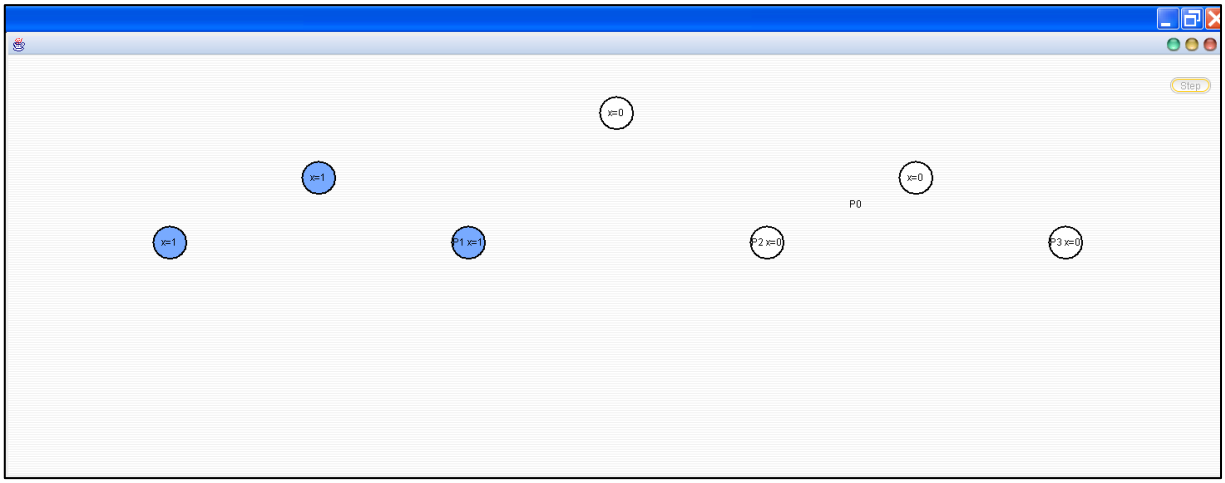
Σχήμα 6.8 – Αλγόριθμος Χ

Ο P0 βρίσκεται σε κόμβο με τιμή ‘1’, έτσι κινείται ένα επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



Σχήμα 6.9 – Αλγόριθμος Χ

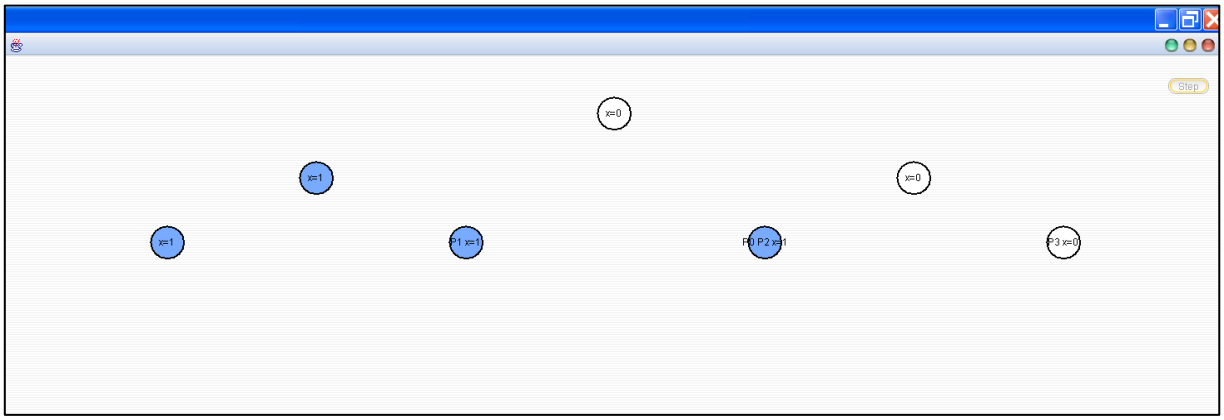
Ο P0 είναι σε κόμβο με τιμή ‘0’, το δεξί παιδί του είναι ‘0’, έτσι κατεβαίνει σε αυτό.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



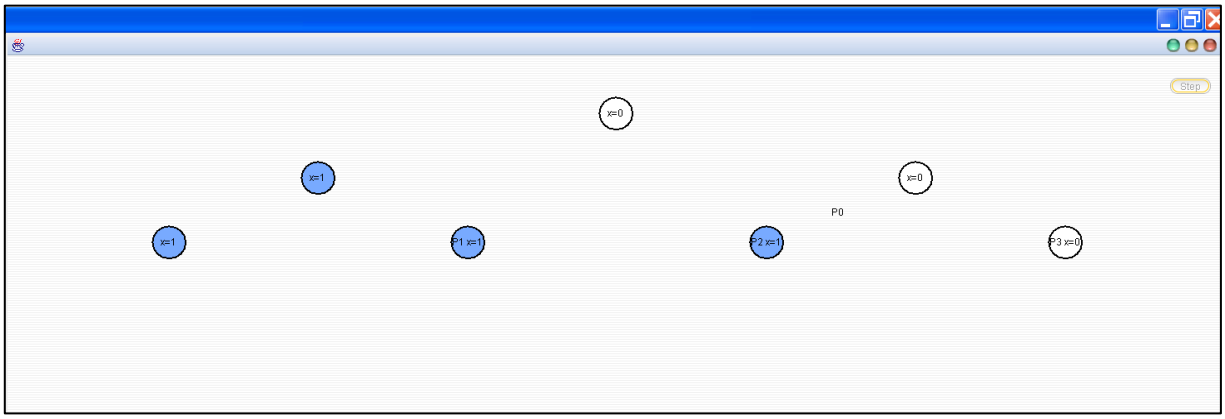
Σχήμα 6.10 – Αλγόριθμος Χ

Ο P0 είναι σε κόμβο με τιμή ‘0’, τα παιδιά του είναι ‘0’, κατεβαίνει στο αριστερό.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

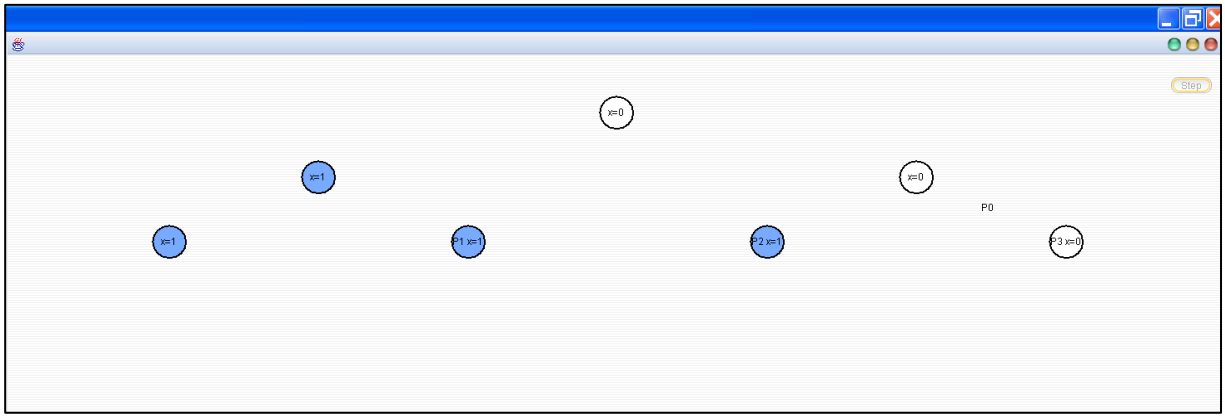
Στο πιο πάνω σχήμα (Σχήμα 6.10) παρατηρούμε ότι ο επεξεργαστής P0 βρίσκεται στον κόμβο 3 που έχει τιμή 0. Αφού έχει τιμή 0, κάνει έλεγχο για την τιμή που έχουν τα παιδιά του, κόμβοι 6 και 7. Τα παιδιά του έχουν και τα δύο την τιμή 0. Για να υπολογίσει σε ποιο από τα δύο θα κατέβει, κάνει τον εξής έλεγχο. Κοιτάζει τον προσωπικό αριθμό του επεξεργαστή και τον μετατρέπει σε binary. Στη συγκεκριμένη περίπτωση, ο επεξεργαστής είναι ο P0 άρα ο προσωπικός του αριθμός είναι 0 και το binary είναι επίσης 0. Στη συνέχεια, κοιτάζει το επίπεδο στο οποίο βρίσκεται, που στην συγκεκριμένη περίπτωση είναι το επίπεδο 1 (τα επίπεδα είναι αριθμημένα, ξεκινώντας από το επίπεδο της ρίζας, που είναι το 0, και κατεβαίνοντας προς τα κάτω) και βάση αυτού εξετάζει το αντίστοιχο bit του αριθμού του επεξεργαστή. Αν είναι 0 κατεβαίνει στο αριστερό παιδί, αν είναι 1 στο δεξί. Στην περίπτωση αυτή, το 1^{ο} bit του αριθμού 0 είναι το 0, άρα κατεβαίνει στο αριστερό παιδί.



Σχήμα 6.11 – Αλγόριθμος X
Ο P0 βρίσκεται σε φύλλο που έχει την τιμή '0', έτσι αλλάζει την τιμή σε '1'.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή '1'.

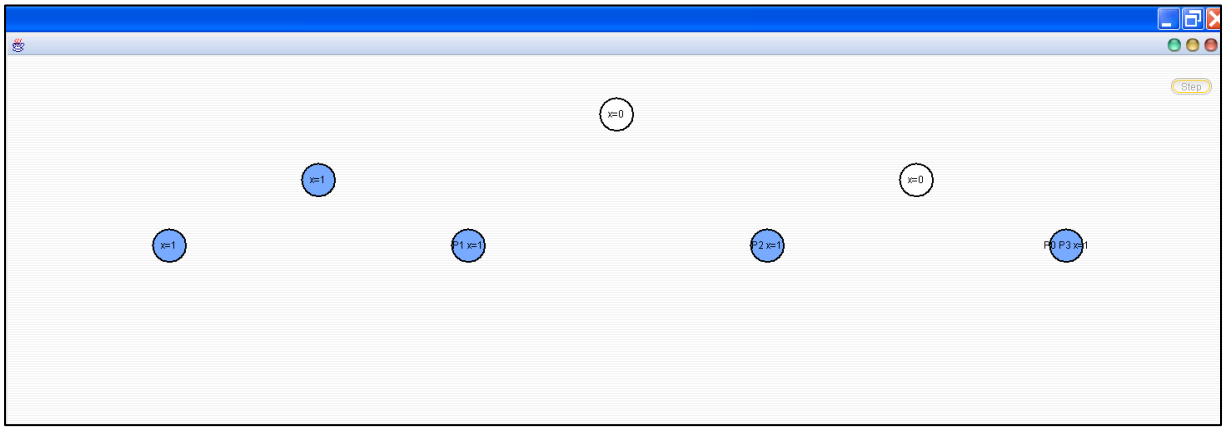


Σχήμα 6.12 – Αλγόριθμος X
Ο P0 βρίσκεται σε φύλλο που έχει την τιμή '1', έτσι κινείται 1 επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή '1'.



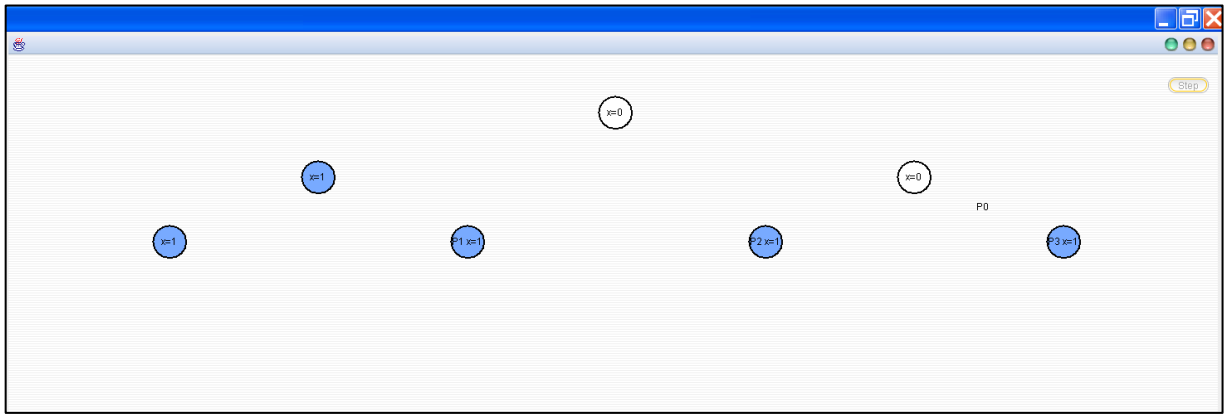
Σχήμα 6.13 – Αλγόριθμος Χ

Ο P0 είναι σε κόμβο με τιμή ‘0’, το δεξί παιδί του είναι ‘0’, έτσι κατεβαίνει σε αυτό.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



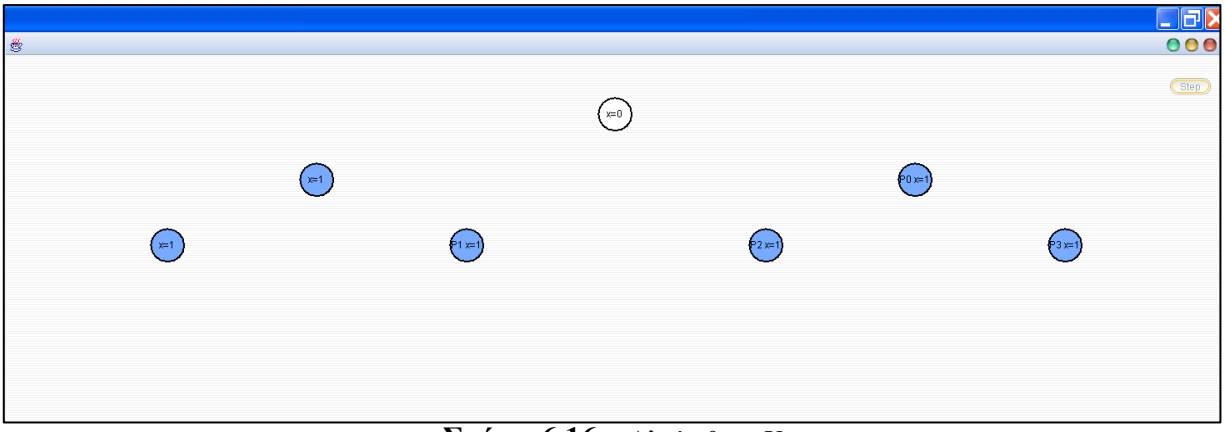
Σχήμα 6.14 – Αλγόριθμος Χ

Ο P0 βρίσκεται σε φύλλο που έχει την τιμή ‘0’, έτσι αλλάζει την τιμή σε ‘1’.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



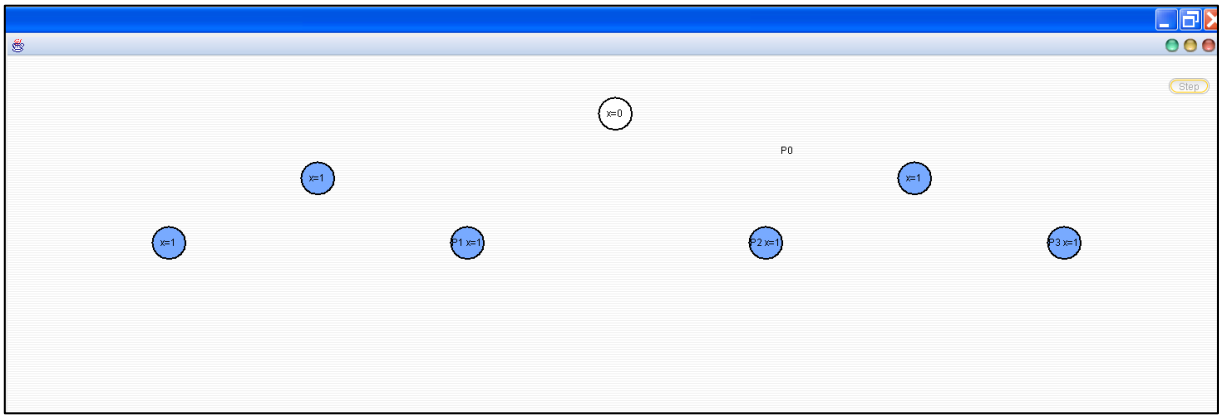
Σχήμα 6.15 – Αλγόριθμος Χ

Ο P0 βρίσκεται σε φύλλο που έχει την τιμή ‘1’, έτσι κινείται 1 επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



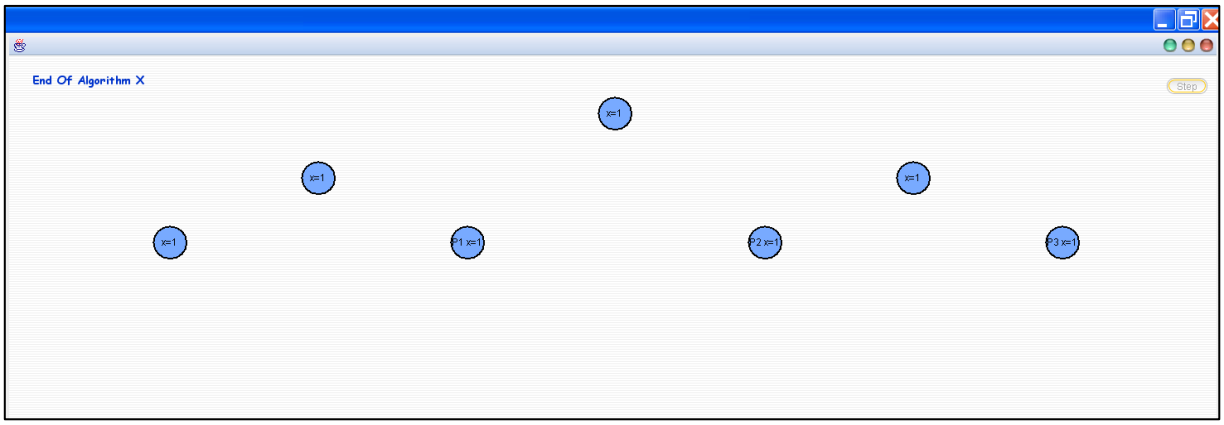
Σχήμα 6.16 – Αλγόριθμος X

Ο P0 βρίσκεται σε κόμβο με τιμή ‘0’, τα παιδιά του έχουν τιμή ‘1’, έτσι γράφει ‘1’.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.



Σχήμα 6.17 – Αλγόριθμος X

Ο P0 βρίσκεται σε κόμβο με τιμή ‘1’, έτσι κινείται ένα επίπεδο προς τα πάνω.
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

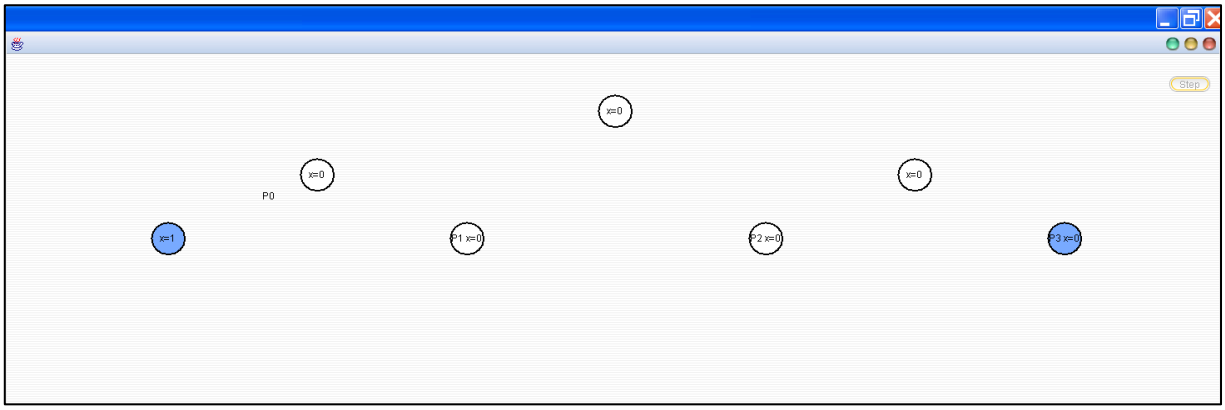


Σχήμα 6.18 – Αλγόριθμος X

Ο P0 βρίσκεται στην ρίζα που έχει τιμή ‘0’, τα παιδιά της είναι ‘1’, έτσι γράφει ‘1’.
Ο αλγόριθμος X έχει τερματίσει με τον επεξεργαστή P0 να φθάνει στην ρίζα, ενώ οι υπόλοιποι επεξεργαστές ακόμα βρίσκονται στην αρχική τους θέση.

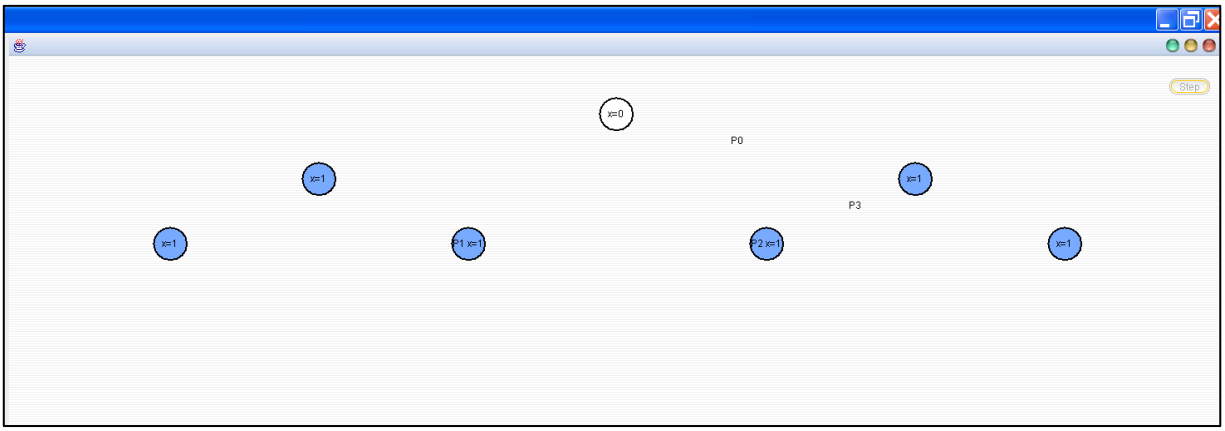
Στο πιο πάνω σχήμα, ο αλγόριθμος Χ έχει τερματιστεί. Στο σενάριο αυτό ο επεξεργαστής P0 είναι πολύ πιο γρήγορος από τους υπόλοιπους επεξεργαστές, με αποτέλεσμα να επισκεφθεί μόνος του όλους τους κόμβους του δέντρου και να γράψει σε αυτούς την τιμή 1, ενώ οι υπόλοιποι επεξεργαστές δεν μετακινήθηκαν καθόλου από τη θέση τους, ούτε έγραψαν την τιμή 1 στα φύλλα που τους είχαν ανατεθεί στην αρχή.

Στα δύο σχήματα που θα ακολουθήσουν, θα παρουσιαστούν δύο στιγμιότυπα από ένα παράδειγμα όπου ο επεξεργαστής P0 είναι ελάχιστα πιο γρήγορος από τον P3, έτσι οι δύο αυτοί επεξεργαστές σε κάποια βήματα εκτελούν πράξεις και οι δύο. Με αυτό τον τρόπο θα δείξουμε δύο περιπτώσεις που δεν είχαμε στο προηγούμενο παράδειγμα. Στην πρώτη περίπτωση ο ένας επεξεργαστής κινείται για να αλλάξει επίπεδο στο δέντρο, ενώ ο άλλος γράφει την τιμή 1 στο φύλλο όπου βρίσκεται και στη δεύτερη κινούνται και οι δύο επεξεργαστές για να αλλάξουν επίπεδο.



Σχήμα 6.19 – Αλγόριθμος Χ
Ο P0 βρίσκεται σε κόμβο με τιμή ‘1’, έτσι κινείται ένα επίπεδο προς τα πάνω.
Ο P3 βρίσκεται σε φύλλο με τιμή ‘0’, έτσι αλλάζει την τιμή σε ‘1’
Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

Στο σχήμα 6.19, παρατηρείται ότι, ενώ το φύλλο όπου βρίσκεται ο επεξεργαστής P3 είναι επιλεγμένο, ακόμα δεν γράφτηκε η τιμή 1. Η τιμή 1 θα γραφτεί αμέσως μόλις φθάσει ο κόμβος P0 στον προορισμό του. Εάν γράφαμε την τιμή 1 αμέσως, τότε το νήμα που τρέχει εκείνη την ώρα για το P3 θα τερματιζόταν και μαζί του θα τερματιζόταν και το νήμα που τρέχει για το P0.



Σχήμα 6.20 – Αλγόριθμος Χ

Οι P0 και P3 βρίσκονται σε κόμβο με τιμή ‘1’, έτσι κινούνται ένα επίπεδο προς τα πάνω. Οι υπόλοιποι επεξεργαστές ακόμα να γράψουν στο φύλλο τους την τιμή ‘1’.

Κεφάλαιο 7

Επίλογος

7.1 Σύνοψη

Η παρούσα διπλωματική εργασία είχε ως σκοπό τη δημιουργία γραφικής διεπαφής με το χρήστη (GUI) και τη γραφική αναπαράσταση παράλληλων εύρωστων αλγορίθμων. Επομένως, στα πλαίσια αυτής της εργασίας, χρειάστηκε αρχικά να μελετηθεί η έννοια του παραλληλισμού και να γίνει αφομοίωση του παράλληλου τρόπου σκέψης για να ξεφύγουμε από το σειριακό. Έτσι μπορέσαμε να κατανοήσουμε καλύτερα τους παράλληλους αλγορίθμους που απασχόλησαν την εργασία αυτή.

Στη συνέχεια, έγινε η μελέτη και η κατανόηση των μοντέλων παράλληλου υπολογισμού, δίνοντας έμφαση στο μοντέλο SIMD (Single Instruction stream, Multiple Data stream) με κοινόχρηστη μνήμη και, πιο συγκεκριμένα, σε ένα μοντέλο που υπάγεται στο SIMD, το PRAM (Parallel Random Access Machine).

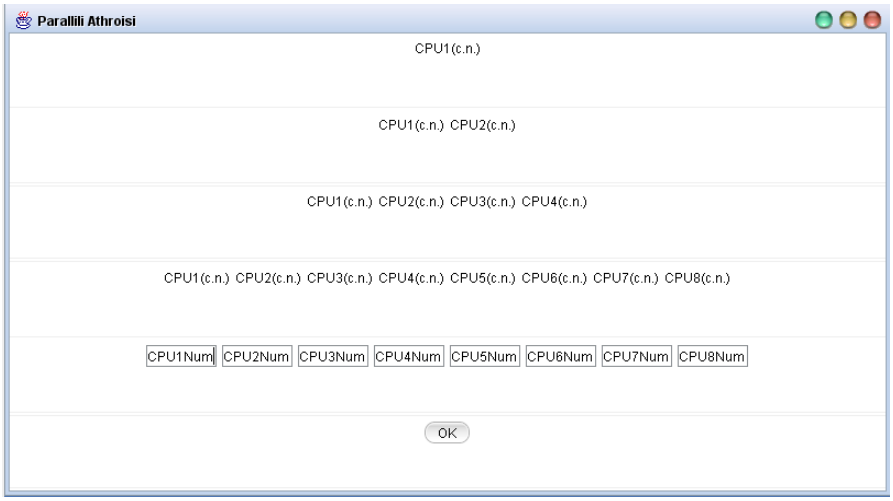
Με το τέλος της μελέτης του μοντέλου PRAM, ξεκίνησε η εκμάθηση του τρόπου λειτουργίας και η μελέτη σημαντικών παράλληλων και εύρωστων αλγορίθμων. Στο σημείο αυτό είχε γίνει και η μελέτη των αλγορίθμων που είχαν απασχολήσει την παρούσα εργασία, δηλαδή ο μη βέλτιστος και ο βέλτιστος αλγόριθμος παράλληλης άθροισης, ο εύρωστος αλγόριθμος W και ο αλγόριθμος X. Κατά τη μελέτη των αλγορίθμων W και X χρειάστηκε να μελετηθούν

και δύο άλλα μοντέλα παράλληλου υπολογισμού, το F-PRAM και το A-PRAM, το οποίο ανήκει στο MIMD, στα οποία βασίζεται η υλοποίηση των αλγορίθμων W και X αντίστοιχα. Γνωρίζοντας πλέον τους αλγορίθμους που είχαμε να υλοποιήσουμε και δημιουργώντας μία αρχική εικόνα του πώς θα θέλαμε να είναι η γραφική αναπαράστασή τους, ξεκινήσαμε την εκμάθηση των εργαλείων (Java Swing, NetBeans IDE, Graphics API) που χρησιμοποιήθηκαν στη δημιουργία των γραφικών αναπαραστάσεων των αλγορίθμων και φυσικά την εις βάθος μελέτη της γλώσσας προγραμματισμού Java (π.χ. χρήση των νημάτων).

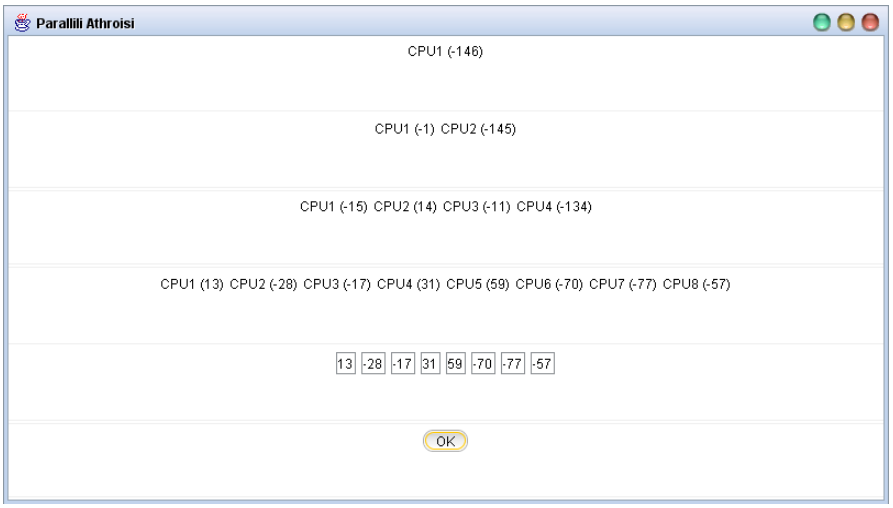
Αφού πρώτα υλοποιήθηκαν κάποια μικρά δοκιμαστικά προγράμματα, μετά δημιουργήθηκε η αρχική φόρμα πάνω στην οποία θα μπορούσαν να τρέχουν οι γραφικές αναπαραστάσεις των αλγορίθμων. Στη συνέχεια, ξεκίνησε η υλοποίηση της γραφικής αναπαράστασης του κάθε αλγορίθμου με την εξής σειρά: πρώτα υλοποιήθηκε ο μη βέλτιστος αλγόριθμος παράλληλης άθροισης, ο οποίος αποτελούσε ένα κομμάτι των επόμενων δύο αλγορίθμων και ακολούθησαν ο βέλτιστος αλγόριθμος παράλληλης άθροισης, ο αλγόριθμος W και τέλος ο αλγόριθμος X.

7.2 Προβλήματα Υλοποίησης

Αρχικά, η ιδέα αυτής της διπλωματικής εργασίας ήταν η υλοποίηση γραφικής αναπαράστασης των αλγορίθμων με την χρήση του Java Swing, χωρίς την χρήση των γραφικών Graphics API της Java. Στην πορεία όμως αφού μελετήσαμε το εργαλείο Java Swing και ξεκίνησε η υλοποίηση του πρώτου αλγορίθμου (μη βέλτιστος αλγόριθμος παράλληλης άθροισης), η δημιουργία δέντρου με τα συστατικά του Java Swing, τοποθετώντας τα δυναμικά με κώδικα ήταν πολύ πολύπλοκη. Ακόμη και έτσι υλοποιήθηκε με τα components του Java Swing η γραφική αναπαράσταση του αλγόριθμου, αλλά η εμφάνιση και η αισθητική που έδινε δεν ήταν τόσο καλή. Επίσης δεν ήταν εφικτή η αναπαράσταση της κίνησης. Έτσι στην πορεία αποφασίστηκε να γίνει η μελέτη και η εκμάθηση του Graphics API της Java και να το χρησιμοποιήσουμε σε συνδυασμό με το Java Swing για να έχουμε και αναπαράσταση κίνησης αλλά και γραφική διεπαφή για τον χρήστη (GUI).



Σχήμα 7.1 – Μη Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Υλοποίηση με την χρήση component του Java Swing (Μέρος 1ο)



Σχήμα 7.2 – Μη Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης Υλοποίηση με την χρήση component του Java Swing (Μέρος 2ο)

Στα πιο πάνω σχήματα (σχήμα 7.1) φαίνετε πώς ήταν η γραφική αναπαράσταση του αλγορίθμου χωρίς την χρήση του Graphics API. Στην φόρμα υπάρχουν τοποθετημένα panels για την δημιουργία των επιπέδων του δέντρου, labels που αντιπροσωπεύουν τους κόμβους του δέντρου και text fields για την καταχώρηση αριθμών στην λίστα από τον χρήστη.

Στην συνέχεια, κατά την υλοποίηση του αλγόριθμου Χ υπήρχαν πρόβλημα με τα threads, που χρησιμοποιήθηκαν με την μέθοδο όπου γίνετε χρήση του Runnable Interface που προσφέρει η Java. Ο λόγος που υπήρχε το πρόβλημα ήταν ότι οι επεξεργαστές στον αλγόριθμο αυτό τρέχουν ασυγχρόνιστα. Έτσι υπήρχε περίπτωση όπου ένας επεξεργαστής, που έστω ονομάζεται P1, χρειαζόταν να αλλάξει επίπεδο στο δέντρο ενώ ένας άλλος, P2, να γράψει την τιμή 1 στην θέση που βρισκόταν στο ίδιο βήμα παράλληλα. Έτσι η γραφική αναπαράσταση της κίνησης του P1 χρειαζόταν περισσότερο χρόνο από ότι ο P2 να γράψει 1. Με αποτέλεσμα

το νήμα που ήταν υπεύθυνο για τον σχεδιασμό της πράξης του P2 να τερματίζει και μαζί του τερμάτιζε και το νήμα για το P1 χωρίς να τελειώσει τον σχεδιασμό της κίνησης. Η λύση που προσφέρθηκε στο πρόβλημα αυτό ήταν να προσθέσουμε κάποια καθυστέρηση στην περίπτωση που είχαμε απλά γραφή σε κάποιο κόμβο της τιμής 1, η οποία είναι εμφανές σε κάποιες περιπτώσεις κατά την διάρκεια της γραφικής αναπαράστασης όπως για παράδειγμα φαίνετε στο Σχήμα 6.19.

Τέλος, να αναφερθούμε στο περιορισμό του μέγιστου αριθμού επεξεργαστών και δεδομένων στους αλγορίθμους. Λόγω του περιορισμένου χώρου που μας δίνεται στην οθόνη δεν μπορούσαμε να υλοποιήσουμε την γραφική αναπαράσταση των αλγορίθμων με μεγαλύτερο n από αυτό που δίνετε στις επιλογές του χρήστη, ανάλογα με τον αλγόριθμο που επιλέγει. Εάν δίναμε την επιλογή για μεγαλύτερο n τότε θα επισκιάζονταν τα δεδομένα του κάθε κόμβου από τον διπλανό του. Το φαινόμενο αυτό μπορούμε να το παρατηρήσουμε ακόμη και τώρα σε μερικές περιπτώσεις στους αλγόριθμους μας συνήθως όταν το n είναι ίσο με 64, δίνουμε όμως την επιλογή αυτή στο χρήστη ως μέγιστη γιατί αν και υπάρχει μία μικρή επισκιάσει των δεδομένων μπορεί να τα διαβάσει.

7.3 Οφέλη Διπλωματικής Εργασίας

Με την εκπόνηση αυτής της διπλωματικής εργασίας τα οφέλη που αποκόμισα ήταν πολλά. Αρχικά, μου δόθηκε η ευκαιρία να κατανοήσω την σωστή μεθοδολογία για την ολοκλήρωση και την συγγραφή μιας εργασίας. Εμπλούτισα τις γνώσεις μου στους αλγορίθμους, προσθέτοντας ένα νέο κεφάλαιο τους παράλληλους αλγορίθμους και ξεχώρισα τις διαφορές μεταξύ σειριακού και παράλληλου υπολογισμού. Με βοήθησε να αντιληφθώ την σημαντικότητα του παραλληλισμού αλλά και τα πλεονεκτήματα του και ενημερώθηκα για τα μοντέλα που υπάρχουν στο παράλληλο υπολογισμό.

Επιπλέον, με βοήθησε να βελτιώσω της προγραμματιστικές μου ικανότητες, να γνωρίσω καλύτερα την γλώσσα προγραμματισμού Java, όπως για παράδειγμα την δημιουργία και χρήση των νημάτων. Επίσης με την μελέτη και εκμάθηση του εργαλείου Java Swing και της κλάσης Graphics API της Java, τα οποία χρειάστηκε να τα μελετήσω και να μάθω την χρήση τους για πρώτη φορά στην διπλωματική αυτή εργασία από μόνη μου, που χρησιμοποιήθηκαν για την γραφική αναπαράσταση των αλγορίθμων, είχα κάποια εμπειρία στην δημιουργία γραφικής διεπαφής (GUI) και στην σχεδίαση με την χρήση γραφικών της Java.

7.4 Μελλοντική Εργασία

Για αυτή την διπλωματική εργασία δημιουργήθηκε γραφική αναπαράσταση για κάποιους παράλληλους αλγόριθμους, όπου ο κάθε ένας επιλύει ένα συγκεκριμένο πρόβλημα. Ο σκοπός υλοποίησης τους, ήταν για να βοηθήσει στην διδασκαλία και στην εκμάθηση της λειτουργίας των αλγορίθμων αυτών, με την χρήση γραφικής αναπαράστασης κάθε βήματος των αλγορίθμων. Κάποιοι αλγόριθμοι αν και πιο εύκολοι στην κατανόηση επιλέχθηκαν λόγω της σημαντικότητας τους. Γιατί η εκμάθησή τους είναι μία σημαντική βάση για να κατανοήσεις τον παραλληλισμό γενικότερα και συνήθως χρησιμοποιούνται από άλλους πιο πολύπλοκους αλγόριθμους, όπως για παράδειγμα ο μη βέλτιστος και ο βέλτιστος αλγόριθμος άθροισης. Κάποιοι άλλοι επιλέχθηκαν λόγο της πολυπλοκότητας τους και των πολλών βημάτων και φάσεων που χρησιμοποιούν μέχρι να επιλύσουν το πρόβλημα, ή ακόμα των απρόβλεπτων σεναρίων που μπορεί να τύχουν, όπως για παράδειγμα ο αλγόριθμος W και ο αλγόριθμος X.

Έτσι αυτό που προτείνω ως μελλοντική εργασία είναι η υλοποίηση γραφικής αναπαράστασης κι άλλων παράλληλων αλγορίθμων που είναι πιο πολύπλοκοι ή αντιμετωπίζεις μεγαλύτερη δυσκολία για την κατανόηση τους. Όπως για παράδειγμα ο αλγόριθμος παράλληλης συγχώνευσης [2,5]. Υπάρχει ένας βαθμός δυσκολίας για την κατανόηση του αλγόριθμου αυτού, για το λόγο ότι είναι δύσκολο και χρονοβόρο να τρέξεις κάποιο παράδειγμα από μόνος σου στο χαρτί. Επίσης στον αλγόριθμο αυτό υπάρχουν αρκετοί έλεγχοι και βήματα τα οποία πρέπει να ακολουθήσεις και αυξάνουν το βαθμό δυσκολίας στην κατανόηση.

Τέλος θα πρότεινα ως μελλοντική εργασία να γίνει προσπάθεια υλοποίησης γραφικής αναπαράστασης των αλγορίθμων που υλοποιήθηκαν σε αυτή την διπλωματική εργασία, χωρίς να υπάρχει περιορισμός στον αριθμό των επεξεργαστών και των δεδομένων. Μετά την υλοποίηση των αλγορίθμων έγινε μία αναζήτηση για να βρεθεί κάποιος τρόπος για να μην υπάρχει πλέον ο περιορισμός. Στην αναζήτηση αυτή βρήκαμε την βιβλιοθήκη Piccolo2D [13], η οποία επιτρέπει την δημιουργία εφαρμογών με γραφικά με τις γλώσσες προγραμματισμού Java και C#, προσθέτοντας οπτικά εφέ όπως zoom (μεγέθυνση) και animation (κίνηση).

Βιβλιογραφία

[1] 2D Graphics (The Java Tutorials), http://download.oracle.com/javase/tutorial/2d/index.html.

[2] Χρύσης Γεωργίου, Σημειώσεις Μαθήματος ΕΠΛ431 – Σύνθεση Παράλληλων Αλγορίθμων, Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου, http://www2.cs.ucy.ac.cy/~chryssis/EPL431/.

[3] Creating a GUI With JFC/Swing (The Java Tutorials), http://download.oracle.com/javase/tutorial/uiswing/index.html.

[4] Bruce Eckel, Thinking in Java, Prentice Hall, Boston, 2006 (4th edition).

[5] Joseph JaJa, An Introduction to Parallel Algorithms, Addison-Wesley, Boston, 1992.

[6] Paris C. Kanellakis and Alex A. Shvartsman, Fault-tolerant Parallel Computation, Kluwer Academic Publishers, 1997.

[7] Τεχνικές Παράλληλου Προγραμματισμού, http://www.it.uom.gr/project/parallel/.

[8] What Is Java technology and why do I need it? http://www.java.com/en/download/faq/whatis_java.xml.

[9] Java SE Documentation, http://download.oracle.com/javase/6/docs/index.html.

[10] NetBeans IDE, http://netbeans.org/community/releases/70/.

[11] Swing (Java™ Foundation Classes), http://download.oracle.com/javase/6/docs/technotes/guides/swing/index.html.

[12] Java2D™ Graphics and Imaging, http://download.oracle.com/javase/6/docs/technotes/guides/2d/index.html.

[13] Piccolo2D, http://www.piccolo2d.org/learn/about.html.

Παράρτημα Α

Κώδικας από Αλγορίθμους που Υλοποιήθηκαν

Α.1 Αλγόριθμος Δημιουργίας Αρχικής Φόρμας (MainWorkspace2.java)

```
/*
 * MainWorkspace2.java
 *
 */


package Gui;
import com.jtattoo.plaf.mcwin.McWinLookAndFeel;
import java.util.Properties;
import javax.swing.DefaultComboBoxModel;
import javax.swing.UIManager;

public class MainWorkspace2 extends javax.swing.JFrame {

    /** Creates new form MainWorkspace2 */
    public MainWorkspace2() {
        try {
            Properties props = new Properties();
            props.put("logoString", "");
            McWinLookAndFeel.setCurrentTheme(props);
       UIManager.setLookAndFeel("com.jtattoo.plaf.mcwin.McWinLookAndFeel");
        } catch (Exception ex) {
        }
        dataSizeModel = new DefaultComboBoxModel();
        cpuSpeedModel = new DefaultComboBoxModel();
        wMode = new DefaultComboBoxModel();
        initComponents();
        cbCpuSpeed.setVisible(false);
        jLabel2.setVisible(false);
        cbMode.setVisible(false);
        jLabel3.setVisible(false);
        cbAnSpeed.setVisible(false);
        jLabel4.setVisible(false);
        cbSearchTypeActionPerformed(null);
    }
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        workspace = new javax.swing.JDesktopPane();
        jPanel1 = new javax.swing.JPanel();
        cbSearchType = new javax.swing.JComboBox();
        jLabel1 = new javax.swing.JLabel();
        btnOK = new javax.swing.JButton();
        cbDataSize = new javax.swing.JComboBox();
        txtInfo = new javax.swing.JScrollPane();
        msgBox = new javax.swing.JTextArea();
        jLabel2 = new javax.swing.JLabel();
        cbCpuSpeed = new javax.swing.JComboBox();
        jLabel3 = new javax.swing.JLabel();
        cbMode = new javax.swing.JComboBox();
        jLabel4 = new javax.swing.JLabel();
        cbAnSpeed = new javax.swing.JComboBox();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```
        workspace.setMinimumSize(new java.awt.Dimension(300, 0));

        jPanel1.setBorder(javax.swing.BorderFactory.createEtchedBorder());

        cbSearchType.setModel(new  javax.swing.DefaultComboBoxModel(new  String[]  {  "Parallel
Adition", "Optimum Parallel Adition", "Algorithm W", "Algorithm X", "Optimum Algorithm W" }));
        cbSearchType.setMinimumSize(new java.awt.Dimension(0, 0));
        cbSearchType.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cbSearchTypeActionPerformed(evt);
            }
        });

        jLabel1.setText("Data size:");

        btnOK.setText("OK");
        btnOK.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnOKActionPerformed(evt);
            }
        });

        cbDataSize.setModel(dataSizeModel);
        cbDataSize.setPreferredSize(new java.awt.Dimension(74, 22));
        cbDataSize.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cbDataSizeActionPerformed(evt);
            }
        });

        msgBox.setColumns(20);
        msgBox.setRows(5);
        txtInfo.setViewportView(msgBox);

        jLabel2.setText("Faster CPU:");

        cbCpuSpeed.setModel(cpuSpeedModel);

        jLabel3.setText("Mode:");

        cbMode.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Custom", "Random"
}));
        cbMode.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                cbModeActionPerformed(evt);
            }
        });

        jLabel4.setText("Animation:");

        cbAnSpeed.setModel(new  javax.swing.DefaultComboBoxModel(new  String[]  {  "Normal",
"Fast", "Instantaneous" }));

        javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
        jPanel1.setLayout(jPanel1Layout);
        jPanel1Layout.setHorizontalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addContainerGap()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel2,    javax.swing.GroupLayout.DEFAULT_SIZE,    76,
Short.MAX_VALUE)
                        .addGap(91, 91, 91))
                    .addGroup(jPanel1Layout.createSequentialGroup()

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 48,
javax.swing.GroupLayout.PREFERRED_SIZE)
                            .addComponent(jLabel1))
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
24, Short.MAX_VALUE)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(cbMode, 0, 86, Short.MAX_VALUE)
                            .addComponent(cbCpuSpeed, 0, 86, Short.MAX_VALUE)
```

```
                            .addComponent(cbDataSize, 0, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)))
                    .addComponent(txtInfo,     javax.swing.GroupLayout.DEFAULT_SIZE,     167,
Short.MAX_VALUE)
                    .addComponent(cbSearchType,  javax.swing.GroupLayout.PREFERRED_SIZE,  167,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel4,   javax.swing.GroupLayout.PREFERRED_SIZE,   77,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                        .addComponent(cbAnSpeed, 0, 85, Short.MAX_VALUE))
                    .addComponent(btnOK, javax.swing.GroupLayout.Alignment.TRAILING))
                .addContainerGap())
        );
        jPanel1Layout.setVerticalGroup(
            jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(90, 90, 90)
                .addComponent(cbSearchType,            javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(70, 70, 70)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(cbMode,              javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel3,     javax.swing.GroupLayout.PREFERRED_SIZE,     19,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(cbDataSize,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(jLabel1,     javax.swing.GroupLayout.PREFERRED_SIZE,     19,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel2,     javax.swing.GroupLayout.PREFERRED_SIZE,     19,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(cbCpuSpeed,          javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(77, 77, 77)
                .addComponent(btnOK)
                .addGap(48, 48, 48)
                .addComponent(txtInfo,        javax.swing.GroupLayout.DEFAULT_SIZE,        101,
Short.MAX_VALUE)
                .addGap(159, 159, 159))
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGap(302, 302, 302)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4,     javax.swing.GroupLayout.PREFERRED_SIZE,     19,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(cbAnSpeed,           javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap(370, Short.MAX_VALUE))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addComponent(jPanel1,        javax.swing.GroupLayout.DEFAULT_SIZE,        190,
Short.MAX_VALUE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(workspace,     javax.swing.GroupLayout.PREFERRED_SIZE,     1593,
javax.swing.GroupLayout.PREFERRED_SIZE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jPanel1,                        javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
            .addComponent(workspace,               javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 698, Short.MAX_VALUE)
        );
```

```
        pack();
    }// </editor-fold>

    private void btnOKActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        switch (cbSearchType.getSelectedIndex()) {
            case 0:{
                int tNum = Integer.parseInt(cbDataSize.getSelectedItem().toString());
                SimpleParallelAdition2 f = new SimpleParallelAdition2(tNum / 2);
                workspace.add(f);
                f.setVisible(true);
                break;
            }
            case 1: {
                int tNum = Integer.parseInt(cbDataSize.getSelectedItem().toString());
                int[] arrayS = new int[tNum];
                for (int i = 0; i < tNum; i++) {
                    arrayS[i] = i;
                }
                OptimumParallelAdition f = new OptimumParallelAdition(tNum, arrayS);
                workspace.add(f);
                f.setVisible(true);
                break;
            }
            case 2: {
                int mode = -1;
                int anSpeed = -1;
                int tempRandomF = cbCpuSpeed.getSelectedIndex();
                int RandomF = 0;
                int tNum = Integer.parseInt(cbDataSize.getSelectedItem().toString());
                mode = cbMode.getSelectedIndex();
                anSpeed = cbAnSpeed.getSelectedIndex();
                if (tempRandomF == 0) {
                    RandomF = 10;
                } else if (tempRandomF == 1) {
                    RandomF = 8;
                } else if (tempRandomF == 2) {
                    RandomF = 6;
                } else if (tempRandomF == 3) {
                    RandomF = 5;
                } else if (tempRandomF == 4) {
                    RandomF = 4;
                } else if (tempRandomF == 5) {
                    RandomF = 3;
                } else if (tempRandomF == 6) {
                    RandomF = 2;
                }
                AlgorithmW f = new AlgorithmW(tNum, mode, anSpeed, RandomF);
                workspace.add(f);
                f.setVisible(true);
                break;
            }
            case 3: {
                int fastCpu = -2;
                int anSpeed = -1;
                int tNum = Integer.parseInt(cbDataSize.getSelectedItem().toString());
                anSpeed = cbAnSpeed.getSelectedIndex();
                if (cbCpuSpeed.getSelectedItem().toString().equals("RANDOM")) {
                    fastCpu = -1;
                } else {
                    fastCpu=
Integer.parseInt(cbCpuSpeed.getSelectedItem().toString().replaceFirst("P", ""));
                }
                AlgorithmX f = new AlgorithmX(tNum, fastCpu, anSpeed);
                workspace.add(f);
                f.setVisible(true);
                break;
            }
            case 4: {
                int mode = -1;
                int anSpeed = -1;
                int tempRandomF = cbCpuSpeed.getSelectedIndex();
                int RandomF = 0;
                int tNum = Integer.parseInt(cbDataSize.getSelectedItem().toString());
                mode = cbMode.getSelectedIndex();
                anSpeed = cbAnSpeed.getSelectedIndex();
                if (tempRandomF == 0) {
                    RandomF = 10;
```

```
                } else if (tempRandomF == 1) {
                    RandomF = 8;
                } else if (tempRandomF == 2) {
                    RandomF = 6;
                } else if (tempRandomF == 3) {
                    RandomF = 5;
                } else if (tempRandomF == 4) {
                    RandomF = 4;
                } else if (tempRandomF == 5) {
                    RandomF = 3;
                } else if (tempRandomF == 6) {
                    RandomF = 2;
                }
                System.out.println(RandomF);
                OptimumAlgorithmW f = new OptimumAlgorithmW(tNum, mode, anSpeed, RandomF);
                workspace.add(f);
                f.setVisible(true);
                break;
            }
        }
}
    private void cbSearchTypeActionPerformed(java.awt.event.ActionEvent evt) {
        switch (cbSearchType.getSelectedIndex()) {
            case 0: {
                jLabel1.setText("Data Size:");
                cbCpuSpeed.setVisible(false);
                jLabel2.setVisible(false);
                cbMode.setVisible(false);
                jLabel3.setVisible(false);
                cbAnSpeed.setVisible(false);
                jLabel4.setVisible(false);
                dataSizeModel.removeAllElements();
                dataSizeModel.addElement("4");
                dataSizeModel.addElement("8");
                dataSizeModel.addElement("16");
                dataSizeModel.addElement("32");
                dataSizeModel.addElement("64");
                break;
            }
            case 1: {
                jLabel1.setText("Data Size:");
                cbCpuSpeed.setVisible(false);
                jLabel2.setVisible(false);
                cbMode.setVisible(false);
                jLabel3.setVisible(false);
                cbAnSpeed.setVisible(false);
                jLabel4.setVisible(false);
                dataSizeModel.removeAllElements();
                dataSizeModel.addElement("4");
                dataSizeModel.addElement("16");
                break;
            }
            case 2: {
                cbCpuSpeed.setModel(wMode);
                jLabel2.setText("Random %");
                jLabel1.setText("Data number:");
                cbCpuSpeed.setVisible(true);
                jLabel2.setVisible(true);
                cbMode.setVisible(true);
                jLabel3.setVisible(true);
                cbAnSpeed.setVisible(true);
                jLabel4.setVisible(true);
                cbModeActionPerformed(null);
                dataSizeModel.removeAllElements();
                cpuSpeedModel.removeAllElements();
                dataSizeModel.addElement("4");
                dataSizeModel.addElement("8");
                dataSizeModel.addElement("16");
                dataSizeModel.addElement("32");
                dataSizeModel.addElement("64");
                break;
            }
            case 3: {
                cbCpuSpeed.setModel(cpuSpeedModel);
                jLabel2.setText("Faster CPU");
                jLabel1.setText("CPU number:");
                cbCpuSpeed.setVisible(true);
                jLabel2.setVisible(true);
```

```
            cbMode.setVisible(false);
            jLabel3.setVisible(false);
            cbAnSpeed.setVisible(true);
            jLabel4.setVisible(true);
            cbDataSizeActionPerformed(null);
            dataSizeModel.removeAllElements();
            cpuSpeedModel.removeAllElements();
            dataSizeModel.addElement("4");
            dataSizeModel.addElement("8");
            dataSizeModel.addElement("16");
            dataSizeModel.addElement("32");
            dataSizeModel.addElement("64");
            break;
        }
        case 4: {
            cbCpuSpeed.setModel(wMode);
            jLabel2.setText("Random %");
            jLabel1.setText("Data number:");
            cbCpuSpeed.setVisible(true);
            jLabel2.setVisible(true);
            cbMode.setVisible(true);
            jLabel3.setVisible(true);
            cbAnSpeed.setVisible(true);
            jLabel4.setVisible(true);
            cbModeActionPerformed(null);
            dataSizeModel.removeAllElements();
            cpuSpeedModel.removeAllElements();
            dataSizeModel.addElement("4");
            dataSizeModel.addElement("16");
            dataSizeModel.addElement("256");
            break;
        }
    }
}

private void cbDataSizeActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    switch (cbDataSize.getSelectedIndex()) {
        case 0: {
            cpuSpeedModel.removeAllElements();
            cpuSpeedModel.addElement("RANDOM");
            for (int i = 0; i < 4; i++) {
                cpuSpeedModel.addElement("P" + i);
            }
            break;
        }
        case 1: {
            cpuSpeedModel.removeAllElements();
            cpuSpeedModel.addElement("RANDOM");
            for (int i = 0; i < 8; i++) {
                cpuSpeedModel.addElement("P" + i);
            }
            break;
        }
        case 2: {
            cpuSpeedModel.removeAllElements();
            cpuSpeedModel.addElement("RANDOM");
            for (int i = 0; i < 16; i++) {
                cpuSpeedModel.addElement("P" + i);
            }
            break;
        }
        case 3: {
            cpuSpeedModel.removeAllElements();
            cpuSpeedModel.addElement("RANDOM");
            for (int i = 0; i < 32; i++) {
                cpuSpeedModel.addElement("P" + i);
            }
            break;
        }
        case 4: {
            cpuSpeedModel.removeAllElements();
            cpuSpeedModel.addElement("RANDOM");
            for (int i = 0; i < 64; i++) {
                cpuSpeedModel.addElement("P" + i);
            }
            break;
        }
```

```
        }
    }

    private void cbModeActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        switch (cbMode.getSelectedIndex()) {
            case 0: {
                cbCpuSpeed.setVisible(false);
                jLabel2.setVisible(false);
                wMode.removeAllElements();
                wMode.addElement("Custom");
                break;
            }
            case 1: {
                cbCpuSpeed.setVisible(true);
                jLabel2.setVisible(true);
                wMode.removeAllElements();
                wMode.addElement("0.1");
                wMode.addElement("0.125");
                wMode.addElement("0.167");
                wMode.addElement("0.2");
                wMode.addElement("0.25");
                wMode.addElement("0.33");
                wMode.addElement("0.5");
                break;
            }
        }
    }
    public static void main(String args[]) {
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new MainWorkspace2().setVisible(true);
            }
        });
    }
    private javax.swing.DefaultComboBoxModel dataSizeModel;
    private javax.swing.DefaultComboBoxModel cpuSpeedModel;
    private javax.swing.DefaultComboBoxModel wMode;
    // Variables declaration - do not modify
    private javax.swing.JButton btnOK;
    private javax.swing.JComboBox cbAnSpeed;
    private javax.swing.JComboBox cbCpuSpeed;
    private javax.swing.JComboBox cbDataSize;
    private javax.swing.JComboBox cbMode;
    private javax.swing.JComboBox cbSearchType;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JTextArea msgBox;
    private javax.swing.JScrollPane txtInfo;
    private javax.swing.JDesktopPane workspace;
    // End of variables declaration
}
```

A.2 Κλάσεις κοινές για όλους τους αλγόριθμους

```
/*
 * FloatingText.java
 *
 */


package Gui;

import java.awt.Graphics;

public class FloatingText {

    public int x, y;
    public String text = "?";
    public boolean visible = true;

    void draw(Graphics g) {
        int c = text.length();
        if (visible) {
            g.drawString(text, x - 3 - (c - 1) * 3, y + 5);
        }
    }
}
```

```
/*
 * MemorySlot.java
 *
 */


package Gui;
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class MemorySlot {

    static Color lightBlue = new Color(120, 170, 255);
    static Color red = new Color(255, 0, 0);
    public int x, y, w, h;
    public String text = "?";
    public boolean selected = false;
    public boolean dead = false;
    public boolean oldSelected = false;
    static BasicStroke penSize = new BasicStroke(2);
    public boolean visible = true;

    void draw(Graphics g) {
        //Draws this processor
        Graphics2D g2 = (Graphics2D) g;
        FontMetrics f = g2.getFontMetrics();
        if (visible) {
            if (selected) {
                g2.setColor(lightBlue);
            } else if (dead) {
                g2.setColor(red);
            } else {
                g2.setColor(Color.white);
            }
            g2.fillRect(x, y, w, h);
            g2.setColor(Color.black);
            g2.drawRect(x, y, w, h);
            g2.drawString(text, x + (w - f.stringWidth(text)) / 2, y + h / 2 + 4);
        }
    }
```

```
    boolean clicked(int mouseX, int mouseY) {
        return (mouseX >= x && mouseX <= x + w && mouseY >= y && mouseY <= y + h);
    }
    int getCenterX() {
        return x + w / 2;
    }
    int getCenterY() {
        return y + h / 2;
    }
}
```

```
/*
 * Processor.java
 *
 */


package Gui;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.FontMetrics;
import java.awt.Graphics;
import java.awt.Graphics2D;

public class Processor {

    static Color lightBlue = new Color(120, 170, 255);
    public int x, y, d;
    int numA = 0, numB = 0;
    public String text = "?";
    public boolean dead = false;
    public boolean selected = false;
    static BasicStroke penSize = new BasicStroke(2);

    void draw(Graphics g) {
        //Draws this processor

        Graphics2D g2 = (Graphics2D) g;
        g2.setStroke(penSize);

        FontMetrics f = g2.getFontMetrics();

        if (selected) {
            g2.setColor(lightBlue);
        } else {
            g2.setColor(Color.white);
        }
        g2.fillOval(x, y, d, d);
        g2.setColor(Color.black);
        g2.drawOval(x, y, d, d);
        g2.drawString(text, x + (d - f.stringWidth(text)) / 2, y + d / 2 + 4);

        if (dead) {
            double r, a;
            int x1, y1, x2, y2;
            r = d / 2;

            for (int i = 0; i < 2; i++) {
                a = Math.PI / 2 * i + Math.PI / 4;
                x1 = (int) (x + r + r * Math.cos(a));
                y1 = (int) (y + r + r * Math.sin(a));
                x2 = (int) (x + r + r * Math.cos(a + Math.PI));
                y2 = (int) (y + r + r * Math.sin(a + Math.PI));
                g2.setColor(Color.red);
                g2.drawLine(x1, y1, x2, y2);
            }
        }
    }

    boolean Clicked(int mouseX, int mouseY) {
        return ((x - mouseX) * (x - mouseX) + (y - mouseY) * (y - mouseY) < d * d / 4);
    }
```

```
    int getCenterX() {
        return x + d / 2;
    }

    int getCenterY() {
        return y + d / 2;
    }

}
```

```
/*
 * DesignLevel.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.util.*;

public class DesignLevel {

    ArrayList<Processor> processors = new ArrayList();
    ArrayList<MemorySlot> memorySlots = new ArrayList();
    ArrayList<FloatingText> floatingText = new ArrayList();
   // int step = 0;
    boolean visible = false;
    boolean animation = false;

    public DesignLevel(int numMemorySlots, int numProcessors, int epipedo, int w, int[] Data)
{

        int levelY = epipedo * 130;

        for (int i = 0; i < numProcessors; i++) {
            Processor p = new Processor();
            p.x = (i * 2 + 1) * w / numMemorySlots;
            p.y = levelY;
            p.d = 40;
            p.text = "";

            processors.add(p);
        }
        for (int i = 0; i < numMemorySlots; i++) {
            MemorySlot m = new MemorySlot();

            m.x = i * w / numMemorySlots + 20;
            m.y = 70 + levelY;
            m.w = w / numMemorySlots - 1;
            m.h = 20;
            m.text = Integer.toString(Data[i]);

            memorySlots.add(m);
        }

        for (int i = 0; i < numProcessors * 2; i++) {
            FloatingText f = new FloatingText();
            f.text = memorySlots.get(i).text;
            floatingText.add(f);
        }
    }

    public void paint(Graphics buffer) {
        if (visible == false) {
            return;
        }

        //Draw all the processors in the list
        for (int i = 0; i < processors.size(); i++) {
            processors.get(i).draw(buffer);
        }
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }
```

```
        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }
    }

    public void update(float ratio, int side) {
        int n = processors.size();
        if (ratio > 0.7 && animation) {
            for (int i = 0; i < n; i++) {
                floatingText.get(i * 2).visible = false;
                floatingText.get(i * 2 + 1).visible = false;
                processors.get(i).text   =   floatingText.get(i   *   2).text   +   "+"   +
floatingText.get(i * 2 + 1).text;
                processors.get(i).numA = Integer.parseInt(floatingText.get(i * 2).text);
                processors.get(i).numB = Integer.parseInt(floatingText.get(i * 2 + 1).text);
            }

            return;
        }

        if (side == 0 || side == 2 && animation) {
            for (int i = 0; i < n; i++) {
                floatingText.get(i * 2).visible = true;
                floatingText.get(i * 2).text=memorySlots.get(i * 2).text;
                floatingText.get(i * 2).x = (int) (memorySlots.get(i * 2).getCenterX() * (1 -
ratio) + processors.get(i).getCenterX() * ratio);
                floatingText.get(i * 2).y = (int) (memorySlots.get(i * 2).getCenterY() * (1 -
ratio) + processors.get(i).getCenterY() * ratio);

            }
        }
        if (side == 1 || side == 2 && animation) {
            for (int i = 0; i < n; i++) {
                floatingText.get(i * 2 + 1).visible = true;
                floatingText.get(i * 2+ 1).text=memorySlots.get(i * 2 + 1).text;
                floatingText.get(i * 2 + 1).x = (int) (memorySlots.get(i * 2 + 1).getCenterX()
* (1 - ratio) + processors.get(i).getCenterX() * ratio);
                floatingText.get(i * 2 + 1).y = (int) (memorySlots.get(i * 2 + 1).getCenterY()
* (1 - ratio) + processors.get(i).getCenterY() * ratio);

            }
        }
    }

    public void updateMemory(float ratio) {
        int n = processors.size();
        if (ratio > 0.8 && animation) {
            for (int i = 0; i < n; i++) {
                floatingText.get(i).visible = false;
                //nextLevel.memorySlots.get(i).text = floatingText.get(i).text;
                memorySlots.get(i).text = floatingText.get(i).text;
                memorySlots.get(i).selected=true;
            }

            return;
        }

        if (animation) {
            for (int i = 0; i < n; i++) {
                floatingText.get(i).text = processors.get(i).text;
                floatingText.get(i).visible = true;
                floatingText.get(i).x = (int) (processors.get(i).getCenterX() * (1 - ratio) +
memorySlots.get(i).getCenterX() * ratio);
                floatingText.get(i).y = (int) (processors.get(i).getCenterY() * (1 - ratio) +
memorySlots.get(i).getCenterY() * ratio);

            }
        }

    }

    public void updateArray(int[] Data, int sizeA){

        for (int i = 0; i < sizeA; i++) {
            Data[i] = Integer.parseInt(memorySlots.get(i).text);
        }

    }
```

```
    public void updateMemorySlotData(int[] Data, int sizeA, DesignLevel nextLevel) {
        for (int i = 0; i < sizeA; i++) {
            nextLevel.memorySlots.get(i).text = Integer.toString(Data[i]);
        }
    }
}
```

```
/*
 * DesignLevelX.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.util.*;

public class DesignLevelX {

    ArrayList<Processor> processors = new ArrayList();
    ArrayList<FloatingText> floatingText = new ArrayList();
   // int step = 0;
    //Thread runner = null;
    boolean animation = false;
    static int temp = 1;
    boolean visible = true;
    int sizeof=0;

    public DesignLevelX(int cpuNum, int currentCpu, int totalEpipeda, int epipedo, int w) {

        int levelY = epipedo * 80;
        sizeof=currentCpu;
        if (totalEpipeda == epipedo) //ean einai stin arxiki fasi sxediasis tou dentrou
        {
            temp = 1;
        }

        for (int i = 0; i < currentCpu; i++) {
            Processor p = new Processor();
            p.x = (i * w / cpuNum + w / cpuNum / 2) * temp;
            p.y = levelY;
            p.d = 40;
            p.text = "x=0";

            processors.add(p);
        }
        temp = temp * 2;

        for (int i = 0; i < currentCpu; i++) {
            FloatingText f = new FloatingText();
            f.text = "P";
            floatingText.add(f);
        }
    }

    public void paint(Graphics buffer) {
        if (visible == false) {
            return;
        }
        //Draw all the processors in the list
        for (int i = 0; i < processors.size(); i++) {
            processors.get(i).draw(buffer);
        }
        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }
    }
}
```

```
/*
 * DesignListOfData.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.util.*;

public class DesignListOfData {
    ArrayList<MemorySlot> memorySlots = new ArrayList();
    boolean animation = false;
    static int temp = 1;
    boolean visible = true;
    int sizeof=0;
    int countWriteAll=0;

    public  DesignListOfData(int  cpuNum,  int  currentCpu,  int  totalEpipeda,  int  w,int
leafListNum) {

        // Dimiourgia listas ari8mwn tou kathe fillou
        sizeof=cpuNum;
        for (int i = 0; i < leafListNum; i++) {
            MemorySlot m = new MemorySlot();
            m.x = (currentCpu * w / cpuNum + w / cpuNum / 2);
            m.y = ((totalEpipeda + 1) * 80) + (i * 20)-35;
            m.w = 40;
            m.h=20;
            m.text = "x=0";
            memorySlots.add(m);
        }
    }

    public void paint(Graphics buffer) {
        if (visible == false) {
            return;
        }
        //Draw all the processors in the list
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }
    }
}
```

A.3 Μη Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης

```
/*
 * SimpleParallelAdition2.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.awt.Image;
import java.util.*;
import java.lang.Math.*;

public class SimpleParallelAdition2 extends javax.swing.JInternalFrame implements Runnable {

    public int lNum = 4;
    public int epipeda = 0;
    int step = 0;
    Thread runner = null;
    int currentLevel = 0;
    ArrayList<DesignLevel> levels = new ArrayList();
    MemorySlot selectedMemorySlot = null;
    int selectedMemorySlotIndex = -1;
    boolean freshlySelected = false;
    boolean startPressed = false;
    int[] dataA;
    int upArray = 1;

    /** Creates new form SimpleParallelAdition2 */
    public SimpleParallelAdition2(int temp) {
        lNum = temp;
        dataA = new int[lNum * 2];
        initComponents();
        epipeda = (int) ((Math.log(lNum) / Math.log(2)) + 1);

        int cpu = lNum;
        int dat = lNum * 2;
        int w = this.getWidth() - 200;

        //Array Initialization
        for (int i = 0; i < dat; i++) {
            dataA[i] = i;
        }

        cpu = lNum;
        for (int j = epipeda; j > 0; j--) {
            levels.add(new DesignLevel(lNum * 2, cpu, j, w, dataA));
            cpu = cpu / 2;
        }
        levels.get(0).visible = true;
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

    …….// kodikas pou dimiourgise to netbeans gia tin dhmiourgia tis formas

    }// </editor-fold>


    private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        //When the user presses the start button reset the positions of the objects
        //and create a timer thread
        int cpuT = lNum;

        startPressed = true;
```

```
        btnStart.setEnabled(false);

        if (upArray == 1) {
            if (selectedMemorySlot != null) {
                selectedMemorySlot.selected = false;
            }
            for (int i = 0; i < lNum * 2; i++) {
                dataA[i] = Integer.parseInt(levels.get(0).memorySlots.get(i).text);
            }
            for (int i = 0; i < levels.size() - 1; i++) {
                for (int j = 0; j < cpuT; j++) {
                    dataA[j] = dataA[2 * j] + dataA[2 * j + 1];
                    levels.get(i + 1).memorySlots.get(j).text = Integer.toString(dataA[j]);
                }
                cpuT = cpuT / 2;
            }
            upArray = 0;
        }
        if (runner == null && currentLevel < epipeda) {
            if (currentLevel < epipeda) {

                if (currentLevel != 0) {
                    levels.get(currentLevel - 1).animation = false;
                    levels.get(currentLevel).visible = true;
                }
                levels.get(currentLevel).animation = true;
                currentLevel++;
            }

            step = 0;
            runner = new Thread(this);
            runner.start();

        }
    }

    private void formMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:

        if (!startPressed) {
            if (selectedMemorySlot != null) {
                if (selectedMemorySlot.text.length() == 0) {
                    selectedMemorySlot.text = "0";
                }
                selectedMemorySlot.selected = false;
                selectedMemorySlot = null;
                selectedMemorySlotIndex = -1;
            }
            for (int i = 0; i < lNum * 2; i++) {
                if (levels.get(0).memorySlots.get(i).clicked(evt.getX(), evt.getY())) {
                    selectedMemorySlot = levels.get(0).memorySlots.get(i);
                    selectedMemorySlot.selected = true;
                    selectedMemorySlotIndex = i;
                    freshlySelected = true;
                }
            }
            repaint();
        }
    }

    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:


        if (!startPressed) {
            if (selectedMemorySlot != null) {
                int c = evt.getKeyCode();
                if (c == 8 && selectedMemorySlot.text.length() > 0) {
                    selectedMemorySlot.text       =       selectedMemorySlot.text.substring(0,
selectedMemorySlot.text.length() - 1);
                    freshlySelected = false;
                } else if (c >= 48 && c <= 57) {
                    if (freshlySelected) {
                        selectedMemorySlot.text = "";
                    }
                    selectedMemorySlot.text += evt.getKeyChar();
                    freshlySelected = false;
                } else if (c == 32) {
```

```
                    selectedMemorySlot.selected = false;
                    selectedMemorySlotIndex = (selectedMemorySlotIndex + 1) % (lNum * 2);
                    selectedMemorySlot                                                    =
levels.get(0).memorySlots.get(selectedMemorySlotIndex);
                    selectedMemorySlot.selected = true;
                    freshlySelected = true;
                }
            }
            repaint();
        }
    }

    private void btnStartKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        formKeyPressed(evt);
    }
    // Variables declaration - do not modify
    private javax.swing.JButton btnStart;
    // End of variables declaration

    public void run() {
        //This function is a thread it's job is to update the
        //possitions of the objects and call the form's paint method
        int state = 0;

        while (runner != null) {
            //update
            if (state == 0) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 100) {
                    state = 1;
                }
                for (int i = 0; i < levels.size(); i++) {
                    levels.get(i).update(a, 2);
                }

            } else if (state == 1) {

                for (int i = 0; i < levels.size(); i++) {
                    for (int j = 0; j < levels.get(i).processors.size(); j++) {
                        int      temp      =      levels.get(i).processors.get(j).numA      +
levels.get(i).processors.get(j).numB;
                        levels.get(i).processors.get(j).text = Integer.toString(temp);
                    }
                }
                state = 2;
                step = 0;

            } else if (state == 2) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 100) {
                    btnStart.setEnabled(true);
                    runner = null;
                }
                for (int i = 0; i < levels.size(); i++) {
                    levels.get(i).updateMemory(a);
                }
            }

            repaint();
            try {
                Thread.sleep(20);
            } catch (InterruptedException e) {
                // do nothing
            }
        }

    }

    @Override
    public void paint(Graphics g) {
        //Called when the form needs to redraw itself
        Image iBuffer;
        Graphics buffer;
```

```
        //Create a buffer
        iBuffer = createImage(this.getWidth(), this.getHeight());
        buffer = iBuffer.getGraphics();

        //Draw the controls and background of the form to the buffer
        super.paintComponents(buffer);
        try {
            for (int i = 0; i < levels.size(); i++) {
                levels.get(i).paint(buffer);
            }
        } catch (Exception ex) {
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);
    }
}
```

A.4 Βέλτιστος Αλγόριθμος Παράλληλης Άθροισης

```
/*
 * OptimumParallelAdition.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.awt.Image;
import java.util.*;
import java.lang.Object.*;

public class OptimumParallelAdition extends javax.swing.JInternalFrame implements Runnable {

    //common
    public int lNum; //cpu num
    int step = 0;
    Thread runner = null;
    MemorySlot selectedMemorySlot = null;
    int selectedMemorySlotIndex = -1;
    boolean startPressed = false;
    boolean freshlySelected = false;
    int[] dataA;
    int part = 0;
    //part1//**************
    ArrayList<Processor> processors = new ArrayList();
    ArrayList<MemorySlot> memorySlots = new ArrayList();
    ArrayList<FloatingText> floatingText = new ArrayList();
    public int dNum;
    public int sData;
    int temp2 = 0;
    //part2//**************
    public int epipeda = 0;
    int currentLevel = 0;
    ArrayList<DesignLevel> levels = new ArrayList();
    int upArray = 1;

    /** Creates new form OptimumParallelAdition */
    public OptimumParallelAdition(int dataNo, int[] arrayS) {
        //part1//**************
        initComponents();

        dNum = dataNo;
        lNum = (int) Math.floor(Math.log(dNum) / Math.log(2));
        sData = dNum / lNum;
        dataA = new int[dNum];
        epipeda = (int) ((Math.log(lNum / 2) / Math.log(2)) + 1);

        for (int i = 0; i < dNum; i++) {
            dataA[i] = arrayS[i];
```

```
    }

    int w = this.getWidth() - 200;

    for (int i = 0; i < lNum; i++) {
        Processor p = new Processor();

        p.x = (i * sData + 1) * w / dNum;
        p.y = 130 * (epipeda + 1);
        p.d = 40;
        p.text = Integer.toString(0);

        processors.add(p);
    }

    for (int i = 0; i < dNum; i++) {
        MemorySlot m = new MemorySlot();

        m.x = i * w / dNum + 20;
        m.y = 70 + ((epipeda + 1) * 130);
        m.w = w / dNum - 1;
        m.h = 20;

        m.text = Integer.toString(dataA[i]);

        memorySlots.add(m);
    }

    for (int i = 0; i < dNum; i++) {
        FloatingText f = new FloatingText();
        f.text = Integer.toString(dataA[i]);
        floatingText.add(f);
    }


    //part2//**************

    int cpu = lNum / 2;

    for (int j = epipeda; j > 0; j--) {
        levels.add(new DesignLevel(dNum, cpu, j, w, dataA));
        cpu = cpu / 2;
    }

}

/** This method is called from within the constructor to
 * initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is
 * always regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
// <editor-fold defaultstate="collapsed" desc="Generated Code">
private void initComponents() {

 …….// kodikas pou dimiourgise to netbeans gia tin dhmiourgia tis formas


}// </editor-fold>

private void formMouseClicked(java.awt.event.MouseEvent evt) {
    // TODO add your handling code here:
    if (!startPressed) {
        if (selectedMemorySlot != null) {
            if (selectedMemorySlot.text.length() == 0) {
                selectedMemorySlot.text = "0";
            }
            selectedMemorySlot.selected = false;
            selectedMemorySlot = null;
            selectedMemorySlotIndex = -1;

        }
        for (int i = 0; i < dNum; i++) {
            if (memorySlots.get(i).clicked(evt.getX(), evt.getY())) {
                selectedMemorySlot = memorySlots.get(i);
                selectedMemorySlot.selected = true;
                selectedMemorySlotIndex = i;
                freshlySelected = true;
```

```
                }
            }
            repaint();
        }
    }

    private void formKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        if (!startPressed) {
            if (selectedMemorySlot != null) {
                int c = evt.getKeyCode();

                if (c == 8 && selectedMemorySlot.text.length() > 0) {
                    selectedMemorySlot.text       =       selectedMemorySlot.text.substring(0,
selectedMemorySlot.text.length() - 1);
                    freshlySelected = false;
                } else if (c >= 48 && c <= 57) {
                    if (freshlySelected) {
                        selectedMemorySlot.text = "";
                    }
                    selectedMemorySlot.text += evt.getKeyChar();
                    freshlySelected = false;
                } else if (c == 32) {
                    selectedMemorySlot.selected = false;
                    selectedMemorySlotIndex = (selectedMemorySlotIndex + 1) % (lNum * 2);
                    selectedMemorySlot = memorySlots.get(selectedMemorySlotIndex);
                    selectedMemorySlot.selected = true;
                    freshlySelected = true;
                }
            }
            repaint();
        }
    }

    private void btnStartKeyPressed(java.awt.event.KeyEvent evt) {
        // TODO add your handling code here:
        formKeyPressed(evt);
    }

    private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        startPressed = true;
        btnStart.setEnabled(false);
        int cpuT = lNum / 2;

        if (part == 0) {
            if (selectedMemorySlot != null) {
                selectedMemorySlot.selected = false;
            }
            for (int i = 0; i < lNum * 2; i++) {
                dataA[i] = Integer.parseInt(memorySlots.get(i).text);

            }
            if (runner == null) {
                step = 0;
                runner = new Thread(this);
                runner.start();

            }
        }
        if (upArray == 1 && part == 1) {

            for (int i = 0; i < dNum; i++) {
                dataA[i] = Integer.parseInt(memorySlots.get(i).text);
                levels.get(0).memorySlots.get(i).text = Integer.toString(dataA[i]);
                levels.get(0).visible = true;
            }
            for (int i = 0; i < levels.size() - 1; i++) {
                for (int j = 0; j < cpuT; j++) {
                    dataA[j] = dataA[2 * j] + dataA[2 * j + 1];
                    levels.get(i + 1).memorySlots.get(j).text = Integer.toString(dataA[j]);

                }

                cpuT = cpuT / 2;

            }
            upArray = 0;
```

```
        }

        if (part == 1) {
            if (runner == null && currentLevel < epipeda) {
                if (currentLevel < epipeda) {

                    if (currentLevel != 0) {
                        levels.get(currentLevel - 1).animation = false;
                        levels.get(currentLevel).visible = true;
                    }
                    levels.get(currentLevel).animation = true;
                    currentLevel++;
                }

                step = 0;
                runner = new Thread(this);
                runner.start();

            }
        }
    }

    public void run() {
        int state = 0;

        while (runner != null) {
            //update
            if (state == 0 && part == 0) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 80) {
                    step = 0;
                    state = 1;
                } else {
                    update(a);
                }
            }
            if (state == 1 && part == 0) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 100) {
                    btnStart.setEnabled(true);
                    part = 1;

                    runner = null;

                } else {
                    updateMemory(a);
                }
            }
            if (state == 0 && part == 1) {
                float a;
                a = step / 100.f;
                step++;
                if (step >= 100) {

                    state = 1;
                }
                for (int i = 0; i < levels.size(); i++) {
                    levels.get(i).update(a, 2);
                }

            } else if (state == 1 && part == 1) {

                for (int i = 0; i < levels.size(); i++) {
                    for (int j = 0; j < levels.get(i).processors.size(); j++) {
                        int      temp      =      levels.get(i).processors.get(j).numA      +
levels.get(i).processors.get(j).numB;
                        levels.get(i).processors.get(j).text = Integer.toString(temp);

                    }
                }
                state = 2;
                step = 0;
            } else if (state == 2 && part == 1) {
                float a;
```

```
            a = step / 100.f;
            step++;
            if (step >= 100) {
                btnStart.setEnabled(true);
                runner = null;

            }
            for (int i = 0; i < levels.size(); i++) {
                levels.get(i).updateMemory(a);

            }

        }

        repaint();
        try {
            Thread.sleep(20);
        } catch (InterruptedException e) {
            // do nothing
        }
    }

}
// Variables declaration - do not modify
private javax.swing.JButton btnStart;
// End of variables declaration

void update(float ratio) {

    int n = processors.size();
    if (ratio > 0.7) {

        try {
            Thread.sleep(250);
        } catch (InterruptedException e) {
            // do nothing
        }

        for (int i = 0; i < n; i++) {
            Processor p = processors.get(i);
            if (temp2 < sData) {
                FloatingText f = floatingText.get((i * sData) + temp2);
                f.visible = false;
                p.text = p.numA + "+" + f.text;
                p.numB = Integer.parseInt(f.text);
                p.numA = p.numA + p.numB;

            }
            if (temp2 == sData) {
                p.text = "s=" + Integer.toString(p.numA);
            }
        }

        temp2++;
        return;
    }

    for (int i = 0; i < memorySlots.size(); i++) {
        Processor p = processors.get((int) Math.floor(i / n));
        MemorySlot m = memorySlots.get(i);
        FloatingText f = floatingText.get(i);
        f.text = m.text;
        f.x = (int) ((p.x + p.d / 2 - 6) * ratio + (m.x + m.w / 2) * (1.f - ratio));
        f.y = (int) ((p.y + p.d / 2 - 6) * ratio + (m.y + 12) * (1.f - ratio));
    }
}
public void updateMemory(float ratio) {
    int n = processors.size();
    if (ratio > 0.8) {
        for (int i = 0; i < n; i++) {
            floatingText.get(i).visible = false;
            memorySlots.get(i).text = floatingText.get(i).text;
            memorySlots.get(i).selected = true;
        }
        return;
    }
    for (int i = 0; i < n; i++) {
        Processor p = processors.get(i);
```

```
            FloatingText f = floatingText.get(i);
            MemorySlot m = memorySlots.get(i);
            f.text = Integer.toString(p.numA);
            f.visible = true;
            f.x = (int) (p.getCenterX() * (1 - ratio) + m.getCenterX() * ratio);
            f.y = (int) (p.getCenterY() * (1 - ratio) + m.getCenterY() * ratio);
        }
    }
    @Override
    public void paint(Graphics g) {
        //Called when the form needs to redraw itself
        Image iBuffer;
        Graphics buffer;

        //Create a buffer
        iBuffer = createImage(this.getWidth(), this.getHeight());
        buffer = iBuffer.getGraphics();

        //Draw the controls and background of the form to the buffer
        super.paintComponents(buffer);

        //Draw all the processors in the list
        for (int i = 0; i < processors.size(); i++) {
            processors.get(i).draw(buffer);
        }
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }
        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);

        try {
            for (int i = 0; i < levels.size(); i++) {
                levels.get(i).paint(buffer);
            }
        } catch (Exception ex) {
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);

    }
}
```

A.5 Αλγόριθμος W

```
/*
 * AlgorithmW.java
 *
 */


package Gui;

import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.util.*;
import java.lang.Math.*;

public class AlgorithmW extends javax.swing.JInternalFrame implements Runnable {

    public int CNum = 0;
    public int epipeda = 0;
    int step = 1;
    int stepAnimation = 0;
    int deadCpu = 0;
```

```
Thread runner = null;
ArrayList<DesignLevelX> levels = new ArrayList(); //dentro proodou
ArrayList<DesignLevelX> levels2 = new ArrayList(); //dentro katametrisis
ArrayList<DesignLevelX> levels3 = new ArrayList(); //dentro 3 phasi 2
ArrayList<MemorySlot> memorySlots = new ArrayList();
ArrayList<FloatingText> floatingText = new ArrayList();
MemorySlot selectedMemorySlot = null;
boolean startPressed = false;
int[][] dataA;//dentro proodou tree 2
int[][] dataB;//dentro katametrisis tree 1
int[][] dataC;//dentro phase 2 tree 3
int totalNodes = 0;
int[][] cpuW;
int phase4counter = 0;
int mode = 0; // 0=custom 1=random
int anSpeed=0; //0=normal 1=fast 2=Instantaneous
int timeSleep=20;
int randomFreq=1000;// dimiourgia pi8anotitas katarrefsis enos cpu

/** Creates new form AlgorithmW */
public AlgorithmW(int temp, int modeTemp, int anSpeedTemp,int randomTemp) {
    CNum = temp;
    mode = modeTemp;
    anSpeed=anSpeedTemp;
    randomFreq=randomTemp;
    int cpu = CNum;
    totalNodes = CNum * 2 - 1;
    dataA = new int[3][totalNodes];// dentro proodou
    dataB = new int[3][totalNodes];//dentro katametrisis
    dataC = new int[3][totalNodes];//dentro 3 temp
    cpuW = new int[4][CNum];

    initComponents();

    epipeda = (int) ((Math.log(CNum) / Math.log(2)) + 1);

    int w = this.getWidth() - 30;

    //Array Initialization
    for (int i = 0; i < totalNodes; i++) {
        dataA[0][i] = 0;
        dataA[1][i] = -1;   //level
        dataA[2][i] = -1;   //index in level
        dataB[0][i] = 0;
        dataB[1][i] = -1;   //level
        dataB[2][i] = -1;   //index in level
        dataC[0][i] = 0;
        dataC[1][i] = -1;   //level
        dataC[2][i] = -1;   //index in level
    }

    int temp2 = totalNodes - CNum;

    for (int i = 0; i < CNum; i++) {
        cpuW[0][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla tou dentrou)
        cpuW[1][i] = 1;  // if the cpu is dead then is 0
        cpuW[2][i] = temp2; //nextPosition
        cpuW[3][i] = 0; //dynamic processor number(phase 1 kai 2)
        temp2++;

    }

    cpu = CNum;
    //Dimiourgia Arxikis thesis ton epe3ergastwn
    for (int i = 0; i < CNum; i++) {
        MemorySlot m = new MemorySlot();

        m.x = i * w / CNum + 20;
        m.y = ((epipeda + 1) * 80);
        m.w = w / CNum - 1;
        m.h = 20;
        m.text = "P" + i;

        memorySlots.add(m);
    }

    for (int i = 0; i < CNum; i++) {
        FloatingText f = new FloatingText();
```

```
            f.text = "P" + i;
            floatingText.add(f);
        }

        //Dimiourgia tou dentrou proodou
        for (int j = epipeda; j > 0; j--) {
            levels.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
            cpu = cpu / 2;
        }
        cpu = CNum;

        //Dimiourgia tou dentrou katametrisis
        for (int j = epipeda; j > 0; j--) {
            levels2.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
            cpu = cpu / 2;
        }
        cpu = CNum;
        //Dimiourgia tou dentrou 3
        for (int j = epipeda; j > 0; j--) {
            levels3.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
            cpu = cpu / 2;
        }
        for (int i = 0; i < epipeda; i++) {
            levels2.get(i).visible = false;
            levels3.get(i).visible = false;
        }
        cpu = 1;
        int k = 0;
        for (int i = epipeda - 1; i >= 0; i--) {
            for (int j = 0; j < cpu; j++) {
                dataA[1][k] = i;
                dataA[2][k] = j;
                dataB[1][k] = i;
                dataB[2][k] = j;
                dataC[1][k] = i;
                dataC[2][k] = j;
                k++;
            }
            cpu = cpu * 2;
        }

        //Setup Animation Speed
        if(anSpeed==0){
            timeSleep=20;
        }else if(anSpeed==1){
            timeSleep=10;
        }else if(anSpeed==2){
            timeSleep=1;
        }

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnStart = new javax.swing.JButton();
        msgLabel = new javax.swing.JLabel();
        msgLabel1 = new javax.swing.JLabel();
        msgLabel2 = new javax.swing.JLabel();

        setClosable(true);
        setIconifiable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("Algorithm W");
        setPreferredSize(new java.awt.Dimension(1478, 866));
        addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                formMouseClicked(evt);
            }
        });
```

```
        btnStart.setText("Step");
        btnStart.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnStartActionPerformed(evt);
            }
        });

        msgLabel.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel.setForeground(new java.awt.Color(0, 51, 204));
        msgLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        msgLabel1.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel1.setForeground(new java.awt.Color(0, 51, 204));
        msgLabel1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        msgLabel2.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel2.setForeground(new java.awt.Color(255, 0, 0));
        msgLabel2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(msgLabel,  javax.swing.GroupLayout.PREFERRED_SIZE,  166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
1219, Short.MAX_VALUE)
                        .addComponent(btnStart))
                    .addComponent(msgLabel1,    javax.swing.GroupLayout.PREFERRED_SIZE,    166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(msgLabel2,    javax.swing.GroupLayout.PREFERRED_SIZE,    238,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(msgLabel,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnStart))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(msgLabel1,      javax.swing.GroupLayout.PREFERRED_SIZE,      38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(msgLabel2,      javax.swing.GroupLayout.PREFERRED_SIZE,      38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(684, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        startPressed = true;
        btnStart.setEnabled(false);
        stepAnimation = 0;
        int cpuPosition;
        int cpuLevel;
        int cpuIndex;
        int randomValue=-1;

        msgLabel2.setText("");

        if(mode==1 && step>1){
            Random generator = new Random();
            for(int i=0;i<CNum;i++){
                if(cpuW[1][i]==1 && deadCpu<CNum-1){
                     randomValue= generator.nextInt(randomFreq);
                     if(randomValue==0){
```

```
                        cpuW[1][i]=0;
                        memorySlots.get(i).dead=true;
                        deadCpu++;
                    }
                }
            }
        }

        for (int i = 0; i < CNum && step < 4; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataA[1][cpuPosition];
                cpuIndex = dataA[2][cpuPosition];
                Processor p = levels.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        for (int i = 0; i < CNum && step > 4 && step < 8; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataB[1][cpuPosition];
                cpuIndex = dataB[2][cpuPosition];
                Processor p = levels2.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        for (int i = 0; i < CNum && step > 8; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataC[1][cpuPosition];
                cpuIndex = dataC[2][cpuPosition];
                Processor p = levels3.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        repaint();

        runner = new Thread(this);
        runner.start();

    }

    private void formMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        if (mode == 0) {
            if (!startPressed && step > 1) {
                for (int i = 0; i < CNum; i++) {
                    if (memorySlots.get(i).clicked(evt.getX(), evt.getY())) {
                        selectedMemorySlot = memorySlots.get(i);
                        if (selectedMemorySlot.dead == true && selectedMemorySlot.oldSelected
== false && deadCpu < CNum) {
                            selectedMemorySlot.dead = false;
                            cpuW[1][i] = 1;
                            deadCpu--;
                        } else if (selectedMemorySlot.oldSelected == false) {
                            if  (deadCpu  >=  CNum  -  1  &&  selectedMemorySlot.oldSelected  ==
false) {
                                System.out.println("At least 1 CPU must stay active!!!!");
                                msgLabel2.setText("At least 1 CPU must stay active!");
                                selectedMemorySlot.dead = false;
                                cpuW[1][i] = 1;
                            } else {
                                selectedMemorySlot.dead = true;
                                cpuW[1][i] = 0;
                                deadCpu++;
                            }
                        }
                    }
                }
                repaint();
            }
        }
```

```
}

public void run() {
    //This function is a thread it's job is to update the
    //possitions of the objects and call the form's paint method
    int state = 0;
    int currentLevel = -1;
    int currentIndex = -1;
    int currentPosition = -1;
    int nextLevel = -1;
    int nextIndex = -1;
    int nextPosition = -1;

    if (step == 1) { //Phase 3
        msgLabel.setText("Tree 2");
        msgLabel1.setText("PHASE W3");
        while (runner != null) {
            float a;
            a = stepAnimation / 100.f;
            stepAnimation++;

            if (stepAnimation >= 72) {
                stepAnimation = 0;
                btnStart.setEnabled(true);
                startPressed = false;
                runner = null;
                step = 2;
            }

            for (int i = 0; i < memorySlots.size(); i++) {
                updateM(a, i, i, "P" + i, 0);
            }

            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
            }

        }
    } else if (step == 2) { //PHASE W3

        msgLabel.setText("Tree 2");
        msgLabel1.setText("PHASE W3");

        for (int i = 0; i < CNum; i++) {
            phase3(i);
        }

        stepAnimation = 0;
        btnStart.setEnabled(true);
        startPressed = false;
        runner = null;
        step = 3;

        repaint();
        try {
            Thread.sleep(timeSleep);
        } catch (InterruptedException e) {
            // do nothing
        }

    } else if (step == 3) { //PHASE w4

        msgLabel.setText("Tree 2");
        msgLabel1.setText("PHASE W4");
        boolean endOfAnimation = true;
        while (runner != null && state == 0) {
            if (state == 0) {

                float a;
                a = stepAnimation / 100.f;
                stepAnimation++;

                if (stepAnimation > 72) {
                    stepAnimation = 0;
                    state = 1;
```

```
                    }
                    for (int i = 0; i < CNum && state == 0; i++) {
                        if (cpuW[1][i] == 1) {
                            currentPosition = cpuW[0][i];
                            if (currentPosition == 0) {
                                state = 2;
                                runner = null;
                                stepAnimation = 0;
                            }
                            currentLevel = dataA[1][currentPosition];
                            currentIndex = dataA[2][currentPosition];
                            nextPosition = (currentPosition + 1) / 2 - 1;
                            nextLevel = dataA[1][nextPosition];
                            nextIndex = dataA[2][nextPosition];
                            Processor                               p                               =
levels.get(currentLevel).processors.get(currentIndex);
                            cpuW[2][i] = nextPosition;
                            if (stepAnimation == 1) {
                                Processor                           p1                              =
levels.get(currentLevel).processors.get(currentIndex);
                                p1.text = p1.text.replaceFirst("P" + i + " ", "");
                            }
                            if (runner != null && state == 0) {
                                endOfAnimation    =    updateP(a,    currentLevel,    nextLevel,
currentIndex, nextIndex, "P" + i, endOfAnimation, i, 0);
                            }
                        }
                    }
                    repaint();
                    try {
                        Thread.sleep(timeSleep);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                }
                if (state == 1) {
                    for (int i = 0; i < CNum; i++) {
                        phase4(i);
                    }

                    try {
                        Thread.sleep(timeSleep*5);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                    repaint();
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    phase4counter++;
                    if (phase4counter < epipeda - 1) {
                        step = 3;
                    } else {
                        if (dataA[0][0] == CNum) {
                            Font font=new Font("Comic Sans MS",Font.BOLD,18);
                            msgLabel2.setFont(font);
                            msgLabel2.setText("End of Algorithm W");
                            btnStart.setEnabled(false);
                            startPressed = true;
                            runner = null;
                        } else {
                            phase4counter = 0;
                            step = 4;
                        }
                    }
                }
            }

        } else if (step == 4) { //Phase W1
            msgLabel.setText("Tree 1");
            msgLabel1.setText("PHASE W1");

            int temp2 = totalNodes - CNum;

            for (int i = 0; i < CNum; i++) { //vazw tou epe3ergastes 3ana sta filla vasi tou
proswpikou tous ari8mou
                cpuW[0][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla tou dentrou)
```

```
            cpuW[2][i] = temp2; //nextPosition
            cpuW[3][i] = 0;       // initialize pn
            temp2++;
        }
        for (int i = 0; i < totalNodes; i++) { //Initialize tree 1 array
            dataB[0][i] = 0;

        }

        //Initialize tou dentrou katametrisis graphics
        for (int j = 0; j < levels2.size(); j++) {
            for (int i = 0; i < levels2.get(j).sizeof; i++) {
                levels2.get(j).processors.get(i).text = "x=0";
            }
        }


        //Allagi dentrou emfanizw to dentro katametrisis
        for (int i = 0; i < epipeda; i++) {
            levels.get(i).visible = false;
            levels2.get(i).visible = true;
        }

        repaint();
        stepAnimation = 0;
        btnStart.setEnabled(true);
        startPressed = false;
        step = 5;
        runner = null;

    } else if (step == 5) {

        while (runner != null) {
            float a;
            a = stepAnimation / 100.f;
            stepAnimation++;

            if (stepAnimation >= 72) {
                stepAnimation = 0;
                btnStart.setEnabled(true);
                startPressed = false;
                runner = null;
                step = 6;
            }

            for (int i = 0; i < memorySlots.size(); i++) {
                if (cpuW[1][i] == 1) {
                    updateM(a, i, i, "P" + i, 1);
                }
            }

            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
            }

        }
    } else if (step == 6) { //Phase W1

        msgLabel.setText("Tree 1");
        msgLabel1.setText("PHASE W1");
        for (int i = 0; i < CNum; i++) {
            phase1(i, 0);
        }
        stepAnimation = 0;
        btnStart.setEnabled(true);
        startPressed = false;
        step = 7;
        repaint();
        try {
            Thread.sleep(timeSleep);
        } catch (InterruptedException e) {
            // do nothing
            }
    } else if (step == 7) { //Phase W1
        boolean endOfAnimation = true;
```

```
            while (runner != null && state == 0) {
                if (state == 0) {

                    float a;
                    a = stepAnimation / 100.f;
                    stepAnimation++;

                    if (stepAnimation > 72) {

                        stepAnimation = 0;
                        state = 1;

                    }
                    for (int i = 0; i < CNum && state == 0; i++) {
                        if (cpuW[1][i] == 1) {
                            currentPosition = cpuW[0][i];
                            if (currentPosition == 0) {
                                state = 2;
                                runner = null;
                                stepAnimation = 0;
                            }
                            currentLevel = dataB[1][currentPosition];
                            currentIndex = dataB[2][currentPosition];
                            nextPosition = (currentPosition + 1) / 2 - 1;
                            nextLevel = dataB[1][nextPosition];
                            nextIndex = dataB[2][nextPosition];
                            Processor                               p                               =
levels2.get(currentLevel).processors.get(currentIndex);
                            cpuW[2][i] = nextPosition;
                            if (stepAnimation == 1) {
                                Processor                           p1                              =
levels2.get(currentLevel).processors.get(currentIndex);
                                p1.text = p1.text.replaceFirst("P" + i + " ", "");
                            }

                            if (runner != null && state == 0) {
                                endOfAnimation    =    updateP(a,    currentLevel,    nextLevel,
currentIndex, nextIndex, "P" + i, endOfAnimation, i, 1);
                            }
                        }
                    }
                    repaint();
                    try {
                        Thread.sleep(timeSleep);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                }
                if (state == 1) {

                    for (int i = 0; i < CNum; i++) {
                        phase1(i, 1);
                    }

                    try {
                        Thread.sleep(timeSleep*5);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                    repaint();
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    phase4counter++;
                    if (phase4counter < epipeda - 1) {
                        step = 7;
                    } else {
                        phase4counter = 0;
                        step = 8;
                    }
                }
            }
        } else if (step == 8) {//Phase W2
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W2");

            //Allagi dentrou emfanizw to dentro 3
```

```
            for (int i = 0; i < epipeda; i++) {
                levels2.get(i).visible = false;
                levels3.get(i).visible = true;
            }
            phase2Tree();
            repaint();
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 9;
            runner = null;

        } else if (step == 9) {//Phase W2
            msgLabel.setText("Tree 3");
            msgLabel1.setText("PHASE W2");

            for (int i = 0; i < CNum; i++) {
                if (cpuW[1][i] == 1) {
                    phase2Part1(i);
                }
            }

            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 10;
            runner = null;

            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
                }

        } else if (step == 10) {// enallagi emfaniseis dentrou prin na pame stin fasi 3
(emfanisi dentrou proodou)
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W3");
            int tempLevel, tempIndex, cpuPosition;
            for (int i = 0; i < totalNodes; i++) {
                tempLevel = dataA[1][i];
                tempIndex = dataA[2][i];
                Processor p = levels.get(tempLevel).processors.get(tempIndex);
                p.text = "x=" + dataA[0][i];
            }
            for (int i = 0; i < CNum; i++) {
                if (cpuW[1][i] == 1) {
                    cpuPosition = cpuW[0][i];
                    tempLevel = dataA[1][cpuPosition];
                    tempIndex = dataA[2][cpuPosition];
                    Processor p = levels.get(tempLevel).processors.get(tempIndex);
                    p.text = "P" + i + " " + p.text;
                }
            }

            for (int i = 0; i < epipeda; i++) {
                levels3.get(i).visible = false;
                levels.get(i).visible = true;
            }
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 2;
            runner = null;
        }

    }

    @Override
    public void paint(Graphics g) {
        //Called when the form needs to redraw itself
        Image iBuffer;
        Graphics buffer;

//Create a buffer
        iBuffer =
                createImage(this.getWidth(), this.getHeight());
```

```
        buffer =
                iBuffer.getGraphics();

        //Draw the controls and background of the form to the buffer
        super.paintComponents(buffer);
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }

        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }

        try {
            for (int i = 0; i <
                    levels.size(); i++) {
                levels.get(i).paint(buffer);
            }
            for (int i = 0; i <
                    levels2.size(); i++) {
                levels2.get(i).paint(buffer);
            }
            for (int i = 0; i <
                    levels3.size(); i++) {
                levels3.get(i).paint(buffer);
            }

        } catch (Exception ex) {
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);
    }

    //floating text animation from "memmory slot" to processor
    void updateM(float ratio, int indexFrom, int indexTo, String data, int treeID) {

        int n = levels.get(0).sizeof;
        if (ratio > 0.7) {

            try {
                Thread.sleep((long) (timeSleep * 3.5));
            } catch (InterruptedException e) {
                // do nothing
            }
            Processor p;
            if (treeID == 0) {
                p = levels.get(0).processors.get(indexTo);
            } else {
                p = levels2.get(0).processors.get(indexTo);
            }
            FloatingText f = floatingText.get(indexFrom);
            f.visible = false;

            p.text = data + " " + p.text;
            return;
        }

        Processor p;
        if (treeID == 0) {
            p = levels.get(0).processors.get(indexTo);
        } else {
            p = levels2.get(0).processors.get(indexTo);
        }

        MemorySlot m = memorySlots.get(indexFrom);
        FloatingText f = floatingText.get(indexFrom);
        f.text = data;
        f.visible = true;

        f.x = (int) ((p.x + p.d / 2) * ratio + (m.x + m.w / 2) * (1.f - ratio));
        f.y = (int) ((p.y + p.d / 2) * ratio + (m.y + m.h / 2) * (1.f - ratio));

    }


    //floating text animation from processor to processor
    boolean updateP(float ratio, int levelFrom, int levelTo, int indexFrom, int indexTo,
String data, boolean endOfAnimation, int CpuID, int treeID) {
```

```
    if (ratio > 0.7) {
        try {
            Thread.sleep((long) (timeSleep * 3.5));
        } catch (InterruptedException e) {
            // do nothing
        }
        Processor p;
        FloatingText f;
        if (treeID == 0) {
            p = levels.get(levelTo).processors.get(indexTo);
            f = levels.get(levelFrom).floatingText.get(indexFrom);
        } else {
            p = levels2.get(levelTo).processors.get(indexTo);
            f = levels2.get(levelFrom).floatingText.get(indexFrom);
        }

        f.visible = false;
        p.text = data + " " + p.text;
        endOfAnimation = false;
        return endOfAnimation;
    }
    Processor p1, p2;
    FloatingText f;
    if (treeID == 0) {
        p1 = levels.get(levelFrom).processors.get(indexFrom);
        p2 = levels.get(levelTo).processors.get(indexTo);
        f = levels.get(levelFrom).floatingText.get(indexFrom);
    } else {
        p1 = levels2.get(levelFrom).processors.get(indexFrom);
        p2 = levels2.get(levelTo).processors.get(indexTo);
        f = levels2.get(levelFrom).floatingText.get(indexFrom);
    }
    f.visible = true;
    f.text = data;
    f.x = (int) ((p2.x + p2.d / 2) * ratio + (p1.x + p1.d / 2) * (1.f - ratio));
    f.y = (int) ((p2.y + p2.d / 2) * ratio + (p1.y + p1.d / 2) * (1.f - ratio));

    return endOfAnimation;
}

//Algorith W-phases
void phase1(int cpuID, int state) {

    int j1, j2, t;//j1,2 siblings and t parent
    if (state == 0) {

        Processor p = levels2.get(0).processors.get(cpuID);
        int currentPosition = cpuW[0][cpuID];
        j1 = currentPosition + 1;
        if (memorySlots.get(cpuID).dead == false) {
            p.text = p.text.replaceFirst("x=0", "x=1");
            dataB[0][currentPosition] = 1;
            cpuW[3][cpuID] = 1;//pn
            t = j1 / 2;
            if (2 * t == j1) {
                j2 = j1 + 1;
            } else {
                j2 = j1 - 1;
            }
            if (dataB[0][j1 - 1] == dataB[0][j2 - 1]) {
                if (j1 > j2) {
                    cpuW[3][cpuID] = cpuW[3][cpuID] + dataB[0][j2 - 1];//pn
                }
            }
        }
    } else {
        if (cpuW[1][cpuID] == 1) {
            int currentPosition = cpuW[2][cpuID];
            j1 = currentPosition + 1;
            cpuW[0][cpuID] = currentPosition;

            int currentLevel = dataB[1][currentPosition];
            int currentIndex = dataB[2][currentPosition];
            int currentData = dataB[0][currentPosition];
            int childRight = dataB[0][(currentPosition + 1) * 2];
            int childLeft = dataB[0][(currentPosition + 1) * 2 - 1];
            int sum = childRight + childLeft;
```

```
                Processor p = levels2.get(currentLevel).processors.get(currentIndex);
                p.text = p.text.replaceFirst("x=" + currentData, "x=" + sum);
                dataB[0][currentPosition] = sum;

                if (j1 > 1) {
                    t = j1 / 2;
                    if (2 * t == j1) {
                        j2 = j1 + 1;
                    } else {
                        j2 = j1 - 1;
                    }
                    if (dataB[0][j1 - 1] == dataB[0][j2 - 1]) {
                        if (j1 > j2) {
                            cpuW[3][cpuID] = cpuW[3][cpuID] + dataB[0][j2 - 1];//pn
                        }
                    }
                }
            }
        }
    }

    void phase2Tree() {
        int tempLevel;
        int tempIndex;
        for (int i = 0; i < totalNodes; i++) {
            dataC[0][i] = dataA[0][i];
            tempLevel = dataC[1][i];
            tempIndex = dataC[2][i];
            Processor p = levels3.get(tempLevel).processors.get(tempIndex);
            p.text = "x=" + dataC[0][i];
        }
    }

    void phase2Part1(int cpuID) {
        int p = 1;//parent
        int tempLevel;
        int tempIndex;
        int size = CNum;
        int temp = -1;
        while (size != 1) {
            tempLevel = dataC[1][p - 1];
            tempIndex = dataC[2][p - 1];
            Processor processor1 = levels3.get(tempLevel).processors.get(tempIndex);
            processor1.selected = true;
            processor1.text = "P" + cpuID + " " + processor1.text;
            int c1 = 2 * p;//left child
            int c2 = c1 + 1;//right child
            //o pateras den einai fillo
            if ((dataA[0][c1 - 1] + dataA[0][c2 - 1]) == 0) {
                temp = dataC[0][c1 - 1];
                dataC[0][c1 - 1] = 0;
                tempLevel = dataC[1][c1 - 1];
                tempIndex = dataC[2][c1 - 1];
                Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
                if (temp != dataC[0][c1 - 1]) {
                    processor.text = "x=" + dataC[0][c1 - 1];
                }
            } else {
                temp = dataC[0][c1 - 1];
                dataC[0][c1 - 1] = dataC[0][p - 1] * dataA[0][c1 - 1] / (dataA[0][c1 - 1] +
dataA[0][c2 - 1]);
                tempLevel = dataC[1][c1 - 1];
                tempIndex = dataC[2][c1 - 1];
                Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
                if (temp != dataC[0][c1 - 1]) {
                    processor.text = "x=" + dataC[0][c1 - 1];
                }
            }
            temp = dataC[0][c2 - 1];
            dataC[0][c2 - 1] = dataC[0][p - 1] - dataC[0][c1 - 1];
            tempLevel = dataC[1][c2 - 1];
            tempIndex = dataC[2][c2 - 1];
            Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
            if (temp != dataC[0][c1 - 1]) {
                processor.text = "x=" + dataC[0][c2 - 1];
            }
```

```
            dataB[0][c1 - 1] = dataB[0][p - 1] * (size / 2 - dataC[0][c1 - 1]) / (size -
dataC[0][p - 1]);
            dataB[0][c2 - 1] = dataB[0][p - 1] - dataB[0][c1 - 1];

            if (cpuW[3][cpuID] <= dataB[0][c1 - 1]) {
                p = c1;
            } else {
                p = c2;
                cpuW[3][cpuID] = cpuW[3][cpuID] - dataB[0][c1 - 1];
            }
            size = size / 2;
            try {
                Thread.sleep(timeSleep*25);
            } catch (InterruptedException e) {
                // do nothing
            }
            repaint();
            processor1.selected = false;
            processor1.text = processor1.text.replaceFirst("P" + cpuID + " ", "");

        }
        if (size == 1) {
            tempLevel = dataC[1][p - 1];
            tempIndex = dataC[2][p - 1];
            Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
            processor.text = "P" + cpuID + " " + processor.text;
            cpuW[0][cpuID] = p - 1;
        }
    }

    void phase3(int cpuID) {
        int currentPosition = cpuW[0][cpuID];
        int tempLevel = dataA[1][currentPosition];
        int tempIndex = dataA[2][currentPosition];
        Processor p = levels.get(tempLevel).processors.get(tempIndex);

        if (cpuW[1][cpuID] == 1) {

            p.text = p.text.replaceFirst("x=0", "x=1");
            dataA[0][currentPosition] = 1;
        }
    }

    void phase4(int cpuID) {

        if (cpuW[1][cpuID] == 1) {

            int currentPosition = cpuW[2][cpuID];
            cpuW[0][cpuID] = currentPosition;
            int currentLevel = dataA[1][currentPosition];
            int currentIndex = dataA[2][currentPosition];
            int currentData = dataA[0][currentPosition];
            int childRight = dataA[0][(currentPosition + 1) * 2];
            int childLeft = dataA[0][(currentPosition + 1) * 2 - 1];
            int sum = childRight + childLeft;
            Processor p = levels.get(currentLevel).processors.get(currentIndex);
            p.text = p.text.replaceFirst("x=" + currentData, "x=" + sum);
            dataA[0][currentPosition] = sum;
        }
    }
    // Variables declaration - do not modify
    private javax.swing.JButton btnStart;
    private javax.swing.JLabel msgLabel;
    private javax.swing.JLabel msgLabel1;
    private javax.swing.JLabel msgLabel2;
    // End of variables declaration
}
```

A.6 Βέλτιστη Εκδοχή Αλγορίθμου W

```
/*
 * OptimumAlgorithmW.java
 *
 */


package Gui;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Image;
import java.util.*;
import java.lang.Math.*;

public class OptimumAlgorithmW extends javax.swing.JInternalFrame implements Runnable {

    public int CNum = 0;
    public int epipeda = 0;
    int step = 0;
    int stepAnimation = 0;
    int deadCpu = 0;
    Thread runner = null;
    ArrayList<DesignLevelX> levels = new ArrayList(); //dentro proodou
    ArrayList<DesignLevelX> levels2 = new ArrayList(); //dentro katametrisis
    ArrayList<DesignLevelX> levels3 = new ArrayList(); //dentro 3 phasi 2
    ArrayList<MemorySlot> memorySlots = new ArrayList();
    // ArrayList<MemorySlot> memorySlots2 = new ArrayList(); //pinakas stoixeiwn kathe fillou
    ArrayList<FloatingText> floatingText = new ArrayList();
    ArrayList<DesignListOfData> listOfData = new ArrayList(); //pinakas stoixeiwn kathe fillou
    MemorySlot selectedMemorySlot = null;
    boolean startPressed = false;
    int[][] dataA;//dentro proodou tree 2
    int[][] dataB;//dentro katametrisis tree 1
    int[][] dataC;//dentro phase 2 tree 3
    int totalNodes = 0;
    int[][] cpuW;
    int phase4counter = 0;
    int mode = 0; // 0=custom 1=random
    int anSpeed = 0; //0=normal 1=fast 2=Instantaneous
    int timeSleep = 20;
    int randomFreq = 1000;// dimiourgia pi8anotitas katarrefsis enos cpu
    int leafListNum = 0;
    int countList = 0;    //metra se poio simeio tis listas vrisketai

    /** Creates new form OptimumAlgorithmW */
    public OptimumAlgorithmW(int temp, int modeTemp, int anSpeedTemp, int randomTemp) {
        CNum = (int) ((int) temp / (Math.floor(Math.log(temp) / Math.log(2))));
        leafListNum = (int) (Math.floor(Math.log(temp) / Math.log(2)));
        countList = leafListNum;
        System.out.println("temp " + temp);
        System.out.println("CNum " + CNum);
        mode = modeTemp;
        anSpeed = anSpeedTemp;
        randomFreq = randomTemp;
        int cpu = CNum;
        totalNodes = CNum * 2 - 1;
        dataA = new int[3][totalNodes];// dentro proodou
        dataB = new int[3][totalNodes];//dentro katametrisis
        dataC = new int[3][totalNodes];//dentro 3 temp
        cpuW = new int[4][CNum];

        initComponents();

        epipeda = (int) ((Math.log(CNum) / Math.log(2)) + 1);

        int w = this.getWidth() - 30;
        int h = this.getHeight() - 30;

        //Array Initialization
        for (int i = 0; i < totalNodes; i++) {
            dataA[0][i] = 0;
            dataA[1][i] = -1;   //level
            dataA[2][i] = -1;   //index in level
            dataB[0][i] = 0;
```

```
    dataB[1][i] = -1;   //level
    dataB[2][i] = -1;   //index in level
    dataC[0][i] = 0;
    dataC[1][i] = -1;   //level
    dataC[2][i] = -1;   //index in level
}

int temp2 = totalNodes - CNum;

for (int i = 0; i < CNum; i++) {
    cpuW[0][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla tou dentrou)
    cpuW[1][i] = 1;  // if the cpu is dead then is 0
    cpuW[2][i] = temp2; //nextPosition
    cpuW[3][i] = 0; //dynamic processor number(phase 1 kai 2)
    temp2++;

}
for (int i = 0; i < CNum; i++) {
    listOfData.add(new DesignListOfData(CNum, i, epipeda, w, leafListNum));

}


cpu = CNum;

//Dimiourgia Arxikis thesis ton epe3ergastwn
for (int i = 0; i < CNum; i++) {
    MemorySlot m = new MemorySlot();

    m.x = i * w / CNum + 20;
    m.y = ((epipeda + 1) * 80) + (leafListNum * 20) - 20;
    m.w = w / CNum - 1;
    m.h = 20;
    m.text = "P" + i;

    memorySlots.add(m);
}

for (int i = 0; i < CNum; i++) {
    FloatingText f = new FloatingText();
    f.text = "P" + i;
    floatingText.add(f);
}

//Dimiourgia tou dentrou proodou
for (int j = epipeda; j > 0; j--) {
    levels.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
    cpu = cpu / 2;
}
cpu = CNum;

//Dimiourgia tou dentrou katametrisis
for (int j = epipeda; j > 0; j--) {
    levels2.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
    cpu = cpu / 2;
}
cpu = CNum;
//Dimiourgia tou dentrou 3
for (int j = epipeda; j > 0; j--) {
    levels3.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
    cpu = cpu / 2;
}
for (int i = 0; i < epipeda; i++) {
    levels2.get(i).visible = false;
    levels3.get(i).visible = false;
}
cpu = 1;
int k = 0;
for (int i = epipeda - 1; i >= 0; i--) {
    for (int j = 0; j < cpu; j++) {
        dataA[1][k] = i;
        dataA[2][k] = j;
        dataB[1][k] = i;
        dataB[2][k] = j;
        dataC[1][k] = i;
        dataC[2][k] = j;
        k++;
    }
```

```
            cpu = cpu * 2;
        }

        //Setup Animation Speed
        if (anSpeed == 0) {
            timeSleep = 20;
        } else if (anSpeed == 1) {
            timeSleep = 10;
        } else if (anSpeed == 2) {
            timeSleep = 1;
        }

    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnStart = new javax.swing.JButton();
        msgLabel = new javax.swing.JLabel();
        msgLabel1 = new javax.swing.JLabel();
        msgLabel2 = new javax.swing.JLabel();

        setClosable(true);
        setIconifiable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("Optimum Algorithm W");
        setPreferredSize(new java.awt.Dimension(1478, 866));
        addMouseListener(new java.awt.event.MouseAdapter() {
            public void mouseClicked(java.awt.event.MouseEvent evt) {
                formMouseClicked(evt);
            }
        });

        btnStart.setText("Step");
        btnStart.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnStartActionPerformed(evt);
            }
        });

        msgLabel.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel.setForeground(new java.awt.Color(0, 51, 204));
        msgLabel.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        msgLabel1.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel1.setForeground(new java.awt.Color(0, 51, 204));
        msgLabel1.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        msgLabel2.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel2.setForeground(new java.awt.Color(255, 0, 0));
        msgLabel2.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 394, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(msgLabel,  javax.swing.GroupLayout.PREFERRED_SIZE,  166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
145, Short.MAX_VALUE)
                        .addComponent(btnStart))
                    .addComponent(msgLabel1,    javax.swing.GroupLayout.PREFERRED_SIZE,    166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(msgLabel2,    javax.swing.GroupLayout.PREFERRED_SIZE,    238,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGap(0, 278, Short.MAX_VALUE)
            .addGroup(layout.createSequentialGroup()
                .addContainerGap()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(msgLabel,     javax.swing.GroupLayout.PREFERRED_SIZE,     38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addComponent(btnStart))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(msgLabel1,      javax.swing.GroupLayout.PREFERRED_SIZE,      38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(msgLabel2,      javax.swing.GroupLayout.PREFERRED_SIZE,      38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(137, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        startPressed = true;
        btnStart.setEnabled(false);
        stepAnimation = 0;
        int cpuPosition;
        int cpuLevel;
        int cpuIndex;
        int randomValue = -1;

        msgLabel2.setText("");

        if (mode == 1 && step > 0) {
            Random generator = new Random();
            for (int i = 0; i < CNum; i++) {
                if (cpuW[1][i] == 1 && deadCpu < CNum - 1) {
                    randomValue = generator.nextInt(randomFreq);
                    if (randomValue == 0) {
                        cpuW[1][i] = 0;
                        memorySlots.get(i).dead = true;
                        deadCpu++;
                    }
                }
            }
        }

        for (int i = 0; i < CNum && step < 4; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataA[1][cpuPosition];
                cpuIndex = dataA[2][cpuPosition];
                Processor p = levels.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        for (int i = 0; i < CNum && step > 4 && step < 8; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataB[1][cpuPosition];
                cpuIndex = dataB[2][cpuPosition];
                Processor p = levels2.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        for (int i = 0; i < CNum && step > 8; i++) {
            if (memorySlots.get(i).dead) {
                memorySlots.get(i).oldSelected = true;
                cpuPosition = cpuW[0][i];
                cpuLevel = dataC[1][cpuPosition];
                cpuIndex = dataC[2][cpuPosition];
```

```
                Processor p = levels3.get(cpuLevel).processors.get(cpuIndex);
                p.text = p.text.replaceFirst("P" + i + " ", "");

            }
        }
        repaint();

        runner = new Thread(this);
        runner.start();
    }

    private void formMouseClicked(java.awt.event.MouseEvent evt) {
        // TODO add your handling code here:
        if (mode == 0) {
            if (!startPressed && step > 0) {
                for (int i = 0; i < CNum; i++) {
                    if (memorySlots.get(i).clicked(evt.getX(), evt.getY())) {
                        selectedMemorySlot = memorySlots.get(i);
                        if (selectedMemorySlot.dead == true && selectedMemorySlot.oldSelected
== false && deadCpu < CNum) {
                            selectedMemorySlot.dead = false;
                            cpuW[1][i] = 1;
                            deadCpu--;
                        } else if (selectedMemorySlot.oldSelected == false) {
                            if (deadCpu >= CNum - 1 && selectedMemorySlot.oldSelected ==
false) {
                                System.out.println("At least 1 CPU must stay active!!!!");
                                msgLabel2.setText("At least 1 CPU must stay active!");
                                selectedMemorySlot.dead = false;
                                cpuW[1][i] = 1;
                            } else {
                                selectedMemorySlot.dead = true;
                                cpuW[1][i] = 0;
                                deadCpu++;
                            }
                        }
                    }
                }
                repaint();
            }
        }
    }

    public void run() {
        //This function is a thread it's job is to update the
        //possitions of the objects and call the form's paint method
        int state = 0;
        int currentLevel = -1;
        int currentIndex = -1;
        int currentPosition = -1;
        int nextLevel = -1;
        int nextIndex = -1;
        int nextPosition = -1;

        if (step == 0) { //Phase 3
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W3");
            while (runner != null) {
                float a;
                a = stepAnimation / 100.f;
                stepAnimation++;

                if (stepAnimation >= 92) {
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    step = 1;
                }

                for (int i = 0; i < memorySlots.size(); i++) {
                    updateM(a, i, i, "P" + i, 0);
                }

                repaint();
                try {
                    Thread.sleep(timeSleep);
                } catch (InterruptedException e) {
```

```
                    // do nothing
                }

            }

        } else if (step == 1) { // Epe3ergasia dedomenwn stin lista  tou ka8e fillou
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W3");


            for (int i = 0; i < CNum; i++) {
                currentPosition = cpuW[0][i];
                int tempIndex = dataA[2][currentPosition];
                MemorySlot   mA   =   listOfData.get(tempIndex).memorySlots.get(leafListNum   -
countList);
                DesignListOfData dL = listOfData.get(tempIndex);
                if (cpuW[1][i] == 1) {
                    mA.selected = true;
                    mA.text = mA.text.replaceFirst("x=0", "x=1");
                    dL.countWriteAll++;


                }
                // mA.selected = false;
            }

            try {
                Thread.sleep(timeSleep * 4);
            } catch (InterruptedException e) {
                // do nothing
            }

            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            runner = null;
            if (countList <= 1) {
                step = 2;
                countList = leafListNum;

            } else {
                countList--;
            }
            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
            }


        } else if (step == 2) { //PHASE W3

            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W3");

            for (int i = 0; i < CNum; i++) {
                phase3(i);
            }

            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            runner = null;
            step = 3;

            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
            }

        } else if (step == 3) { //PHASE w4

            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W4");
            boolean endOfAnimation = true;
```

```
            while (runner != null && state == 0) {
                if (state == 0) {

                    float a;
                    a = stepAnimation / 100.f;
                    stepAnimation++;

                    if (stepAnimation > 72) {
                        stepAnimation = 0;
                        state = 1;
                    }
                    for (int i = 0; i < CNum && state == 0; i++) {
                        if (cpuW[1][i] == 1) {
                            currentPosition = cpuW[0][i];
                            if (currentPosition == 0) {
                                state = 2;
                                runner = null;
                                stepAnimation = 0;
                            }
                            currentLevel = dataA[1][currentPosition];
                            currentIndex = dataA[2][currentPosition];
                            nextPosition = (currentPosition + 1) / 2 - 1;
                            nextLevel = dataA[1][nextPosition];
                            nextIndex = dataA[2][nextPosition];
                            Processor                                p                                =
levels.get(currentLevel).processors.get(currentIndex);
                            cpuW[2][i] = nextPosition;
                            if (stepAnimation == 1) {
                                Processor                            p1                               =
levels.get(currentLevel).processors.get(currentIndex);
                                p1.text = p1.text.replaceFirst("P" + i + " ", "");
                            }
                            if (runner != null && state == 0) {
                                endOfAnimation    =    updateP(a,    currentLevel,    nextLevel,
currentIndex, nextIndex, "P" + i, endOfAnimation, i, 0);
                            }
                        }
                    }
                    repaint();
                    try {
                        Thread.sleep(timeSleep);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                }
                if (state == 1) {
                    for (int i = 0; i < CNum; i++) {
                        phase4(i);
                    }

                    try {
                        Thread.sleep(timeSleep * 5);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                    repaint();
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    phase4counter++;
                    if (phase4counter < epipeda - 1) {
                        step = 3;
                    } else {
                        if (dataA[0][0] == CNum) {
                            Font font = new Font("Comic Sans MS", Font.BOLD, 18);
                            msgLabel2.setFont(font);
                            msgLabel2.setText("End of Algorithm W");
                            btnStart.setEnabled(false);
                            startPressed = true;
                            runner = null;
                        } else {
                            phase4counter = 0;
                            step = 4;
                        }
                    }
                }
            }
```

```
        } else if (step == 4) { //Phase W1
            msgLabel.setText("Tree 1");
            msgLabel1.setText("PHASE W1");

            int temp2 = totalNodes - CNum;

            for (int i = 0; i < CNum; i++) { //vazw tou epe3ergastes 3ana sta filla vasi tou
proswpikou tous ari8mou
                cpuW[0][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla tou dentrou)
                cpuW[2][i] = temp2; //nextPosition
                cpuW[3][i] = 0;       // initialize pn
                temp2++;
            }
            for (int i = 0; i < totalNodes; i++) { //Initialize tree 1 array
                dataB[0][i] = 0;

            }

            //Initialize tou dentrou katametrisis graphics
            for (int j = 0; j < levels2.size(); j++) {
                for (int i = 0; i < levels2.get(j).sizeof; i++) {
                    levels2.get(j).processors.get(i).text = "x=0";
                }
            }


            //Allagi dentrou emfanizw to dentro katametrisis
            for (int i = 0; i < epipeda; i++) {
                levels.get(i).visible = false;
                levels2.get(i).visible = true;
            }

            // To dentro katametrisis den exei pinakes sta filla tou. Fevgw tous pinakes

            for (int j = 0; j < listOfData.size(); j++) {
                DesignListOfData dL = listOfData.get(j);
                dL.visible = false;
            }


            repaint();
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 5;
            runner = null;

        } else if (step == 5) {

            while (runner != null) {
                float a;
                a = stepAnimation / 100.f;
                stepAnimation++;

                if (stepAnimation >= 92) {
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    step = 6;
                }

                for (int i = 0; i < memorySlots.size(); i++) {
                    if (cpuW[1][i] == 1) {
                        updateM(a, i, i, "P" + i, 1);
                    }
                }

                repaint();
                try {
                    Thread.sleep(timeSleep);
                } catch (InterruptedException e) {
                    // do nothing
                }

            }
        } else if (step == 6) { //Phase W1
```

```
            msgLabel.setText("Tree 1");
            msgLabel1.setText("PHASE W1");
            for (int i = 0; i < CNum; i++) {
                phase1(i, 0);
            }
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 7;
            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
                }
        } else if (step == 7) { //Phase W1
            boolean endOfAnimation = true;
            while (runner != null && state == 0) {
                if (state == 0) {

                    float a;
                    a = stepAnimation / 100.f;
                    stepAnimation++;

                    if (stepAnimation > 72) {

                        stepAnimation = 0;
                        state = 1;

                    }
                    for (int i = 0; i < CNum && state == 0; i++) {
                        if (cpuW[1][i] == 1) {
                            currentPosition = cpuW[0][i];
                            if (currentPosition == 0) {
                                state = 2;
                                runner = null;
                                stepAnimation = 0;
                            }
                            currentLevel = dataB[1][currentPosition];
                            currentIndex = dataB[2][currentPosition];
                            nextPosition = (currentPosition + 1) / 2 - 1;
                            nextLevel = dataB[1][nextPosition];
                            nextIndex = dataB[2][nextPosition];
                            Processor                               p                               =
levels2.get(currentLevel).processors.get(currentIndex);
                            cpuW[2][i] = nextPosition;
                            if (stepAnimation == 1) {
                                Processor                           p1                              =
levels2.get(currentLevel).processors.get(currentIndex);
                                p1.text = p1.text.replaceFirst("P" + i + " ", "");
                            }

                            if (runner != null && state == 0) {
                                endOfAnimation    =    updateP(a,    currentLevel,    nextLevel,
currentIndex, nextIndex, "P" + i, endOfAnimation, i, 1);
                            }
                        }
                    }
                    repaint();
                    try {
                        Thread.sleep(timeSleep);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                }
                if (state == 1) {

                    for (int i = 0; i < CNum; i++) {
                        phase1(i, 1);
                    }

                    try {
                        Thread.sleep(timeSleep * 5);
                    } catch (InterruptedException e) {
                        // do nothing
                    }
                    repaint();
```

```
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    startPressed = false;
                    runner = null;
                    phase4counter++;
                    if (phase4counter < epipeda - 1) {
                        step = 7;
                    } else {
                        phase4counter = 0;
                        step = 8;
                    }
                }
            }
        } else if (step == 8) {//Phase W2
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W2");

            //Allagi dentrou emfanizw to dentro 3
            for (int i = 0; i < epipeda; i++) {
                levels2.get(i).visible = false;
                levels3.get(i).visible = true;
            }

            //emfanizw to pinaka me ta stoixeia twn fillwn
            for (int j = 0; j < listOfData.size(); j++) {
                DesignListOfData dL = listOfData.get(j);
                dL.visible = true;
            }

            phase2Tree();
            repaint();
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 9;
            runner = null;

        } else if (step == 9) {//Phase W2
            msgLabel.setText("Tree 3");
            msgLabel1.setText("PHASE W2");

            for (int i = 0; i < CNum; i++) {
                if (cpuW[1][i] == 1) {
                    phase2Part1(i);
                }
            }

            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 10;
            runner = null;

            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
                }

        } else if (step == 10) {// enallagi emfaniseis dentrou prin na pame stin fasi 3
(emfanisi dentrou proodou)
            msgLabel.setText("Tree 2");
            msgLabel1.setText("PHASE W3");
            int tempLevel, tempIndex, cpuPosition;
            for (int i = 0; i < totalNodes; i++) {
                tempLevel = dataA[1][i];
                tempIndex = dataA[2][i];
                Processor p = levels.get(tempLevel).processors.get(tempIndex);
                p.text = "x=" + dataA[0][i];
            }
            for (int i = 0; i < CNum; i++) {
                if (cpuW[1][i] == 1) {
                    cpuPosition = cpuW[0][i];
                    tempLevel = dataA[1][cpuPosition];
                    tempIndex = dataA[2][cpuPosition];
                    Processor p = levels.get(tempLevel).processors.get(tempIndex);
                    p.text = "P" + i + " " + p.text;
```

```
                }
            }

            for (int i = 0; i < epipeda; i++) {
                levels3.get(i).visible = false;
                levels.get(i).visible = true;
            }
            stepAnimation = 0;
            btnStart.setEnabled(true);
            startPressed = false;
            step = 1;
            runner = null;
        }

    }

    @Override
    public void paint(Graphics g) {
        //Called when the form needs to redraw itself
        Image iBuffer;
        Graphics buffer;

//Create a buffer
        iBuffer =
                createImage(this.getWidth(), this.getHeight());
        buffer =
                iBuffer.getGraphics();

        //Draw the controls and background of the form to the buffer
        super.paintComponents(buffer);
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }

        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }

        try {
            for (int i = 0; i <
                    levels.size(); i++) {
                levels.get(i).paint(buffer);
            }
            for (int i = 0; i <
                    levels2.size(); i++) {
                levels2.get(i).paint(buffer);
            }
            for (int i = 0; i <
                    levels3.size(); i++) {
                levels3.get(i).paint(buffer);
            }
            for (int i = 0; i < listOfData.size(); i++) {
                listOfData.get(i).paint(buffer);
            }

        } catch (Exception ex) {
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);
    }

    //floating text animation from "memmory slot" to processor
    void updateM(float ratio, int indexFrom, int indexTo, String data, int treeID) {

        int n = levels.get(0).sizeof;
        if (ratio > 0.9) {

            try {
                Thread.sleep((long) (timeSleep * 3.5));
            } catch (InterruptedException e) {
                // do nothing
            }
            Processor p;
            if (treeID == 0) {
                p = levels.get(0).processors.get(indexTo);
            } else {
                p = levels2.get(0).processors.get(indexTo);
            }
```

```
            FloatingText f = floatingText.get(indexFrom);
            f.visible = false;

            p.text = data + " " + p.text;
            return;
        }

        Processor p;
        if (treeID == 0) {
            p = levels.get(0).processors.get(indexTo);
        } else {
            p = levels2.get(0).processors.get(indexTo);
        }

        MemorySlot m = memorySlots.get(indexFrom);
        FloatingText f = floatingText.get(indexFrom);
        f.text = data;
        f.visible = true;

        f.x = (int) ((p.x + p.d / 2) * ratio + (m.x + m.w / 2) * (1.f - ratio));
        f.y = (int) ((p.y + p.d / 2) * ratio + (m.y + m.h / 2) * (1.f - ratio));

    }

    //floating text animation from processor to processor
    boolean updateP(float ratio, int levelFrom, int levelTo, int indexFrom, int indexTo,
String data, boolean endOfAnimation, int CpuID, int treeID) {

        if (ratio > 0.7) {
            try {
                Thread.sleep((long) (timeSleep * 3.5));
            } catch (InterruptedException e) {
                // do nothing
            }
            Processor p;
            FloatingText f;
            if (treeID == 0) {
                p = levels.get(levelTo).processors.get(indexTo);
                f = levels.get(levelFrom).floatingText.get(indexFrom);
            } else {
                p = levels2.get(levelTo).processors.get(indexTo);
                f = levels2.get(levelFrom).floatingText.get(indexFrom);
            }

            f.visible = false;
            p.text = data + " " + p.text;
            endOfAnimation = false;
            return endOfAnimation;
        }
        Processor p1, p2;
        FloatingText f;
        if (treeID == 0) {
            p1 = levels.get(levelFrom).processors.get(indexFrom);
            p2 = levels.get(levelTo).processors.get(indexTo);
            f = levels.get(levelFrom).floatingText.get(indexFrom);
        } else {
            p1 = levels2.get(levelFrom).processors.get(indexFrom);
            p2 = levels2.get(levelTo).processors.get(indexTo);
            f = levels2.get(levelFrom).floatingText.get(indexFrom);
        }
        f.visible = true;
        f.text = data;
        f.x = (int) ((p2.x + p2.d / 2) * ratio + (p1.x + p1.d / 2) * (1.f - ratio));
        f.y = (int) ((p2.y + p2.d / 2) * ratio + (p1.y + p1.d / 2) * (1.f - ratio));

        return endOfAnimation;
    }

    //Algorith W-phases
    void phase1(int cpuID, int state) {

        int j1, j2, t;//j1,2 siblings and t parent
        if (state == 0) {

            Processor p = levels2.get(0).processors.get(cpuID);
            int currentPosition = cpuW[0][cpuID];
            j1 = currentPosition + 1;
            if (memorySlots.get(cpuID).dead == false) {
```

```
            p.text = p.text.replaceFirst("x=0", "x=1");
            dataB[0][currentPosition] = 1;
            cpuW[3][cpuID] = 1;//pn
            t = j1 / 2;
            if (2 * t == j1) {
                j2 = j1 + 1;
            } else {
                j2 = j1 - 1;
            }
            if (dataB[0][j1 - 1] == dataB[0][j2 - 1]) {
                if (j1 > j2) {
                    cpuW[3][cpuID] = cpuW[3][cpuID] + dataB[0][j2 - 1];//pn
                }
            }
        }
    } else {
        if (cpuW[1][cpuID] == 1) {
            int currentPosition = cpuW[2][cpuID];
            j1 = currentPosition + 1;
            cpuW[0][cpuID] = currentPosition;

            int currentLevel = dataB[1][currentPosition];
            int currentIndex = dataB[2][currentPosition];
            int currentData = dataB[0][currentPosition];
            int childRight = dataB[0][(currentPosition + 1) * 2];
            int childLeft = dataB[0][(currentPosition + 1) * 2 - 1];
            int sum = childRight + childLeft;
            Processor p = levels2.get(currentLevel).processors.get(currentIndex);
            p.text = p.text.replaceFirst("x=" + currentData, "x=" + sum);
            dataB[0][currentPosition] = sum;

            if (j1 > 1) {
                t = j1 / 2;
                if (2 * t == j1) {
                    j2 = j1 + 1;
                } else {
                    j2 = j1 - 1;
                }
                if (dataB[0][j1 - 1] == dataB[0][j2 - 1]) {
                    if (j1 > j2) {
                        cpuW[3][cpuID] = cpuW[3][cpuID] + dataB[0][j2 - 1];//pn
                    }
                }
            }
        }
    }
}

void phase2Tree() {
    int tempLevel;
    int tempIndex;
    for (int i = 0; i < totalNodes; i++) {
        dataC[0][i] = dataA[0][i];
        tempLevel = dataC[1][i];
        tempIndex = dataC[2][i];
        Processor p = levels3.get(tempLevel).processors.get(tempIndex);
        p.text = "x=" + dataC[0][i];
    }
}

void phase2Part1(int cpuID) {
    int p = 1;//parent
    int tempLevel;
    int tempIndex;
    int size = CNum;
    int temp = -1;
    while (size != 1) {
        tempLevel = dataC[1][p - 1];
        tempIndex = dataC[2][p - 1];
        Processor processor1 = levels3.get(tempLevel).processors.get(tempIndex);
        processor1.selected = true;
        processor1.text = "P" + cpuID + " " + processor1.text;
        int c1 = 2 * p;//left child
        int c2 = c1 + 1;//right child
        //o pateras den einai fillo
        if ((dataA[0][c1 - 1] + dataA[0][c2 - 1]) == 0) {
            temp = dataC[0][c1 - 1];
            dataC[0][c1 - 1] = 0;
```

```
                tempLevel = dataC[1][c1 - 1];
                tempIndex = dataC[2][c1 - 1];
                Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
                if (temp != dataC[0][c1 - 1]) {
                    processor.text = "x=" + dataC[0][c1 - 1];
                }
            } else {
                temp = dataC[0][c1 - 1];
                dataC[0][c1 - 1] = dataC[0][p - 1] * dataA[0][c1 - 1] / (dataA[0][c1 - 1] +
dataA[0][c2 - 1]);
                tempLevel = dataC[1][c1 - 1];
                tempIndex = dataC[2][c1 - 1];
                Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
                if (temp != dataC[0][c1 - 1]) {
                    processor.text = "x=" + dataC[0][c1 - 1];
                }
            }
            temp = dataC[0][c2 - 1];
            dataC[0][c2 - 1] = dataC[0][p - 1] - dataC[0][c1 - 1];
            tempLevel = dataC[1][c2 - 1];
            tempIndex = dataC[2][c2 - 1];
            Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
            if (temp != dataC[0][c1 - 1]) {
                processor.text = "x=" + dataC[0][c2 - 1];
            }

            dataB[0][c1 - 1] = dataB[0][p - 1] * (size / 2 - dataC[0][c1 - 1]) / (size -
dataC[0][p - 1]);
            dataB[0][c2 - 1] = dataB[0][p - 1] - dataB[0][c1 - 1];

            if (cpuW[3][cpuID] <= dataB[0][c1 - 1]) {
                p = c1;
            } else {
                p = c2;
                cpuW[3][cpuID] = cpuW[3][cpuID] - dataB[0][c1 - 1];
            }
            size = size / 2;
            try {
                Thread.sleep(timeSleep * 25);
            } catch (InterruptedException e) {
                // do nothing
            }
            repaint();
            processor1.selected = false;
            processor1.text = processor1.text.replaceFirst("P" + cpuID + " ", "");

        }
        if (size == 1) {
            tempLevel = dataC[1][p - 1];
            tempIndex = dataC[2][p - 1];
            Processor processor = levels3.get(tempLevel).processors.get(tempIndex);
            processor.text = "P" + cpuID + " " + processor.text;
            cpuW[0][cpuID] = p - 1;
        }
    }

    void phase3(int cpuID) {
        int currentPosition = cpuW[0][cpuID];
        int tempLevel = dataA[1][currentPosition];
        int tempIndex = dataA[2][currentPosition];
        Processor p = levels.get(tempLevel).processors.get(tempIndex);

        if (cpuW[1][cpuID] == 1) {

            p.text = p.text.replaceFirst("x=0", "x=1");
            dataA[0][currentPosition] = 1;
        }
    }

    void phase4(int cpuID) {

        if (cpuW[1][cpuID] == 1) {

            int currentPosition = cpuW[2][cpuID];
            cpuW[0][cpuID] = currentPosition;
            int currentLevel = dataA[1][currentPosition];
            int currentIndex = dataA[2][currentPosition];
            int currentData = dataA[0][currentPosition];
```

```
            int childRight = dataA[0][(currentPosition + 1) * 2];
            int childLeft = dataA[0][(currentPosition + 1) * 2 - 1];
            int sum = childRight + childLeft;
            Processor p = levels.get(currentLevel).processors.get(currentIndex);
            p.text = p.text.replaceFirst("x=" + currentData, "x=" + sum);
            dataA[0][currentPosition] = sum;
        }
    }
    // Variables declaration - do not modify
    private javax.swing.JButton btnStart;
    private javax.swing.JLabel msgLabel;
    private javax.swing.JLabel msgLabel1;
    private javax.swing.JLabel msgLabel2;
    // End of variables declaration
}
```

A.7 Αλγόριθμος X

```
/*
 * AlgorithmX.java
 *
 */


package Gui;

import java.awt.Graphics;
import java.awt.Image;
import java.util.*;
import java.lang.Math.*;

public class AlgorithmX extends javax.swing.JInternalFrame implements Runnable {

    public int CNum = 0;
    public int epipeda = 0;
    int step = 0;
    int round = 0;
    int stepAnimation = 0;
    Thread runner = null;
    ArrayList<DesignLevelX> levels = new ArrayList();
    ArrayList<MemorySlot> memorySlots = new ArrayList();
    ArrayList<FloatingText> floatingText = new ArrayList();
    boolean startPressed = false;
    int[][] dataA;// = new int[lNum * 2];
    int[][] cpuSpeed;
    int upArray = 1;
    int totalNodes = 0;
    Vector<Integer> vCPUReady = new Vector<Integer>();
    private Object lock_1 = new int[1];
    int anSpeed=0; //0=normal 1=fast 2=Instantaneous
    int timeSleep=20;


    /** Creates new form AlgorithmX */
    public AlgorithmX(int temp, int fasterCpu, int anSpeedTemp) {
        CNum = temp;
        int cpu = CNum;
        anSpeed=anSpeedTemp;
        totalNodes = CNum * 2 - 1;
        dataA = new int[3][totalNodes];
        cpuSpeed = new int[4][CNum];

        initComponents();
        epipeda = (int) ((Math.log(CNum) / Math.log(2)) + 1);

        int w = this.getWidth() - 30;
```

```
    //Array Initialization
    for (int i = 0; i < totalNodes; i++) {
        dataA[0][i] = 0;
        dataA[1][i] = -1;   //level
        dataA[2][i] = -1;   //index in level

    }
    int temp2 = totalNodes - CNum;

    for (int i = 0; i < CNum; i++) {
        cpuSpeed[0][i] = 0; //cpuSpeed
        cpuSpeed[1][i] = 0; //cpuLastRoundUsed
        cpuSpeed[2][i] = temp2; //cpuLastNodeVisit(arxika vriskete sta filla tou dentrou)
        cpuSpeed[3][i] = 1;  // if the cpu comes to an end the value becomes 0
        temp2++;
    }

    Random generator = new Random();

    for (int i = 0; i < CNum; i++) {
        int r = generator.nextInt(100) + 1;
        cpuSpeed[0][i] = r;
        cpuSpeed[1][i] = r;
    }

    if (fasterCpu != -1) {
        cpuSpeed[0][fasterCpu] = 1;
        cpuSpeed[1][fasterCpu] = 1;
    }

    cpu = CNum;
    //Dimiourgia Arxikis thesis ton epe3ergastwn
    for (int i = 0; i < CNum; i++) {
        MemorySlot m = new MemorySlot();

        m.x = i * w / CNum + 20;
        m.y = ((epipeda + 1) * 80);
        m.w = w / CNum - 1;
        m.h = 20;

        m.text = "P" + i;

        memorySlots.add(m);
    }

    for (int i = 0; i < CNum; i++) {
        FloatingText f = new FloatingText();
        f.text = "P" + i;
        floatingText.add(f);
    }

    //Dimiourgia tou dentrou
    for (int j = epipeda; j > 0; j--) {
        levels.add(new DesignLevelX(CNum, cpu, epipeda, j, w));
        cpu = cpu / 2;
    }

    cpu = 1;
    int k = 0;
    for (int i = epipeda - 1; i >= 0; i--) {
        for (int j = 0; j < cpu; j++) {
            dataA[1][k] = i;
            dataA[2][k] = j;
            k++;
        }
        cpu = cpu * 2;
    }
     //Setup Animation Speed
    if(anSpeed==0){
        timeSleep=20;
    }else if(anSpeed==1){
        timeSleep=10;
    }else if(anSpeed==2){
        timeSleep=1;
    }

}
```

```
    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        btnStart = new javax.swing.JButton();
        msgLabel = new javax.swing.JLabel();

        setClosable(true);
        setIconifiable(true);
        setMaximizable(true);
        setResizable(true);
        setTitle("Algorithm X");

        btnStart.setText("Step");
        btnStart.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                btnStartActionPerformed(evt);
            }
        });

        msgLabel.setFont(new java.awt.Font("Comic Sans MS", 1, 14));
        msgLabel.setForeground(new java.awt.Color(0, 51, 204));
        msgLabel.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap()
                .addComponent(msgLabel,      javax.swing.GroupLayout.PREFERRED_SIZE,      166,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,     1219,
Short.MAX_VALUE)
                .addComponent(btnStart)
                .addContainerGap())
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(msgLabel,          javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
                    .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createSequentialGroup()
                        .addGap(28, 28, 28)
                        .addComponent(btnStart)))
                .addContainerGap(776, Short.MAX_VALUE))
        );

        pack();
    }// </editor-fold>

    private void btnStartActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
        startPressed = true;
        btnStart.setEnabled(false);
        stepAnimation = 0;

        step++;
        int minSpeed = Integer.MAX_VALUE;

        int end = 0;

        if (step > 1) {
            for (int i = 0; i < CNum; i++) {
                if (cpuSpeed[1][i] < minSpeed && cpuSpeed[3][i] == 1) {
                    minSpeed = cpuSpeed[1][i];
```

```
                }
            }
            for (int i = 0; i < CNum; i++) {
                if (cpuSpeed[1][i] == minSpeed && cpuSpeed[3][i] == 1) {
                    vCPUReady.add(i);
                    runner = new Thread(this);
                    runner.start();
                    cpuSpeed[1][i] += cpuSpeed[0][i];
                    end++;
                }
            }
            if (end == 0) {
                btnStart.setEnabled(false);
                msgLabel.setText("End Of Algorithm X");
            }

            round += minSpeed;
        } else {
            runner = new Thread(this);
            runner.start();
        }

}
    // Variables declaration - do not modify
    private javax.swing.JButton btnStart;
    private javax.swing.JLabel msgLabel;
    // End of variables declaration

    public void run() {
        //This function is a thread it's job is to update the
        //possitions of the objects and call the form's paint method

        int CPUID;
        int levelFrom = 0;
        int levelTo = 0;
        int indexFrom = 0;
        int indexTo = 0;
        int currentPosition = 0;
        int nextPosition = 0;
        boolean isLeaf = false;
        boolean endOfAnimation = true;

        if (step == 1) {
            while (runner != null) {

                float a;
                a = stepAnimation / 100.f;
                stepAnimation++;

                if (stepAnimation >= 72) {
                    for (int j = 0; j < memorySlots.size(); j++) {
                        memorySlots.get(j).visible = false;
                    }
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                    runner = null;
                }

                for (int i = 0; i < memorySlots.size(); i++) {
                    updateM(a, i, i, "P" + i);
                }

                repaint();
                try {
                    Thread.sleep(timeSleep);
                } catch (InterruptedException e) {
                    // do nothing
                }
            }
        } else {

            synchronized (lock_1) {
                CPUID = vCPUReady.get(0);
                vCPUReady.remove(0);
            }

            currentPosition = cpuSpeed[2][CPUID];
            levelFrom = dataA[1][currentPosition];
```

```
indexFrom = dataA[2][currentPosition];
nextPosition = calculateNextPosition(currentPosition, CPUID, levelFrom);

Processor p = levels.get(levelFrom).processors.get(indexFrom);

if (nextPosition != -1 && nextPosition != -2) {
    levels.get(levelFrom).visible = true;
    levelTo = dataA[1][nextPosition];
    indexTo = dataA[2][nextPosition];
} else if (nextPosition == -1 && dataA[0][currentPosition] == 1) {
    p.selected = true;

    for (int i = 0; i < 50; i++) {
        try {
            Thread.sleep((long) (timeSleep * 1.75));
        } catch (InterruptedException e) {
            //do nothing
        }
    }

    p.text = p.text.replaceFirst("x=0", "x=1");
    repaint();

    for (int i = 0; i < 60; i++) {
        try {
            Thread.sleep(timeSleep*3);
        } catch (InterruptedException e) {
            // do nothing
        }
    }

    if (currentPosition == 0 && nextPosition == -1) {
        msgLabel.setText("Processor " + CPUID + " ended!!!");

        for (int i = 0; i < 60; i++) {
            try {
                Thread.sleep((long) (timeSleep * 1.5));
            } catch (InterruptedException e) {
                // do nothing
            }
        }

        msgLabel.setText("End Of Algorithm X");
        btnStart.setEnabled(false);
        runner = null;
    } else {
        nextPosition = currentPosition;
        isLeaf = true;
        stepAnimation = 0;
        btnStart.setEnabled(true);
        runner = null;
    }
} else if (nextPosition == -2) {
    msgLabel.setText("Processor " + CPUID + " ended!!!");
    nextPosition = currentPosition;

    cpuSpeed[3][CPUID] = 0;
    stepAnimation = 0;
    repaint();
    try {
        Thread.sleep(timeSleep*5);
    } catch (InterruptedException e) {
        // do nothing
        }
    btnStart.setEnabled(false);
    msgLabel.setText("End Of Algorithm X");
    runner = null;

}

if (!(p.text.equals("x=0")) && !isLeaf) {
    if (!(p.text.equals("x=1"))) {

        p.text = p.text.replaceFirst("P" + CPUID + " ", "");
    }
}

cpuSpeed[2][CPUID] = nextPosition;
```

```
            while (runner != null) {
                float a;
                a = stepAnimation / 100.f;
                stepAnimation++;
                if (stepAnimation > 100) {
                    runner = null;
                    stepAnimation = 0;
                    btnStart.setEnabled(true);
                }

                if (runner != null) {
                    endOfAnimation = updateP(a, levelFrom, levelTo, indexFrom, indexTo, "P" +
CPUID, endOfAnimation, CPUID);
                }

                repaint();
                try {
                    Thread.sleep(timeSleep);
                } catch (InterruptedException e) {
                    // do nothing
                }
            }
            repaint();
            try {
                Thread.sleep(timeSleep);
            } catch (InterruptedException e) {
                // do nothing
            }
        }
    }

    @Override
    public void paint(Graphics g) {
        //Called when the form needs to redraw itself
        Image iBuffer;
        Graphics buffer;

        //Create a buffer
        iBuffer = createImage(this.getWidth(), this.getHeight());
        buffer = iBuffer.getGraphics();

        //Draw the controls and background of the form to the buffer
        super.paintComponents(buffer);
        for (int i = 0; i < memorySlots.size(); i++) {
            memorySlots.get(i).draw(buffer);
        }

        for (int i = 0; i < floatingText.size(); i++) {
            floatingText.get(i).draw(buffer);
        }

        try {
            for (int i = 0; i < levels.size(); i++) {
                levels.get(i).paint(buffer);
            }

        } catch (Exception ex) {
        }
        //Copy the buffer to the screen
        g.drawImage(iBuffer, 0, 0, this);
    }

    void updateM(float ratio, int indexFrom, int indexTo, String data) {

        int n = levels.get(0).sizeof;
        if (ratio > 0.7) {

            try {
                Thread.sleep((long) (timeSleep * 3.5));
            } catch (InterruptedException e) {
                // do nothing
            }

            Processor p = levels.get(0).processors.get(indexTo);
            FloatingText f = floatingText.get(indexFrom);
            f.visible = false;
            p.text = data + " " + p.text;
```

```
            return;
        }

        Processor p = levels.get(0).processors.get(indexTo);
        MemorySlot m = memorySlots.get(indexFrom);
        FloatingText f = floatingText.get(indexFrom);
        f.text = data;

        f.x = (int) ((p.x + p.d / 2) * ratio + (m.x + m.w / 2) * (1.f - ratio));
        f.y = (int) ((p.y + p.d / 2) * ratio + (m.y + m.h / 2) * (1.f - ratio));

    }

    boolean updateP(float ratio, int levelFrom, int levelTo, int indexFrom, int indexTo,
String data, boolean endOfAnimation, int CpuID) {

        if (ratio > 0.7) {
            try {
                Thread.sleep(timeSleep*5);
            } catch (InterruptedException e) {
                // do nothing
            }

            Processor p = levels.get(levelTo).processors.get(indexTo);
            FloatingText f = levels.get(levelFrom).floatingText.get(indexFrom);
            f.visible = false;

            float temp3 = 0;
            if (CpuID > 53) {
                temp3 = (float) (0.7 + ((float) CpuID / 300));
            } else if (CpuID > 26) {
                temp3 = (float) (0.7 + ((float) CpuID / 200));
            } else {
                temp3 = (float) (0.7 + ((float) CpuID / 100));
            }

            if (endOfAnimation && ratio > temp3) {
                p.text = data + " " + p.text;
                endOfAnimation = false;
            }
            return endOfAnimation;
        }

        Processor p1 = levels.get(levelFrom).processors.get(indexFrom);
        Processor p2 = levels.get(levelTo).processors.get(indexTo);
        MemorySlot m = memorySlots.get(indexFrom);
        FloatingText f = levels.get(levelFrom).floatingText.get(indexFrom);
        f.visible = true;
        f.text = data;

        f.x = (int) ((p2.x + p2.d / 2) * ratio + (p1.x + p1.d / 2) * (1.f - ratio));
        f.y = (int) ((p2.y + p2.d / 2) * ratio + (p1.y + p1.d / 2) * (1.f - ratio));

        return endOfAnimation;
    }

    //Algorith X
    int calculateNextPosition(int currentPosition, int cpuID, int levelFrom) {
        int nextPosition = -1;
        int where = currentPosition + 1;
        int bitNum = epipeda - levelFrom;
        int even = cpuID & bitNum;

        if (dataA[0][currentPosition] == 1) {
            if (currentPosition != 0) {
                nextPosition = (where / 2) - 1;
            } else {
                nextPosition = -2;     //Vriskete stin Riza kai einai 1
            }

        } else if (dataA[0][currentPosition] != 1 && where > CNum - 1) { //Vriskete se Fillo
            dataA[0][currentPosition] = 1;
            //  nextPosition=(where/2)-1;  //anevainei epipedo
        } else if (dataA[0][currentPosition] != 1 && where <= CNum - 1) { // Den vriskete se
Fillo
            if (dataA[0][2 * where - 1] == 1 && dataA[0][2 * where] == 1) { //Ta 2 paidia tou
komvou einai 1
                dataA[0][currentPosition] = 1;
```

```
                //  nextPosition=(where/2)-1; //anevainei epipedo
            } else if (dataA[0][2 * where - 1] == 1 && dataA[0][2 * where] == 0) { //To paidi
2n+1 einai 0
                nextPosition = 2 * where; //katevainei epipedo
            } else if (dataA[0][2 * where - 1] == 0 && dataA[0][2 * where] == 1) { //To 1
paidi 2n einai 0
                nextPosition = 2 * where - 1; //katevainei epipedo
            } else if (dataA[0][2 * where - 1] == 0 && dataA[0][2 * where] == 0) { //Kai ta 2
paidia einai 0
                if ((even == 1)) {  // Cpu last bit is 1
                    nextPosition = 2 * where;
                } else {
                    nextPosition = 2 * where - 1; //katevainei epipedo
                }
            }
        }

        return nextPosition;
    }
}
```