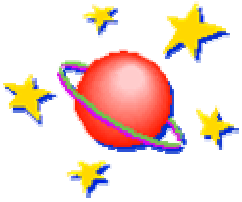


Ενότητα 8η : *Χρήση μεταγλωττιστή για την εκμετάλλευση της παραλληλίας σε επίπεδο εντολής*

Σκοπός **Ο σκοπός της ενότητας αυτής είναι να παρουσιάσει τη χρησιμότητα του μεταγλωττιστή για την εκμετάλλευση της παραλληλίας σε επίπεδο εντολής. Επίσης, στην ενότητα αυτή θα εξετάσουμε τεχνικές για την ανίχνευση των εξαρτήσεων και την εξάλειψη των εξαρτήσεων ονομάτων. Τέλος, θα γνωρίσουμε τη σωλήνωση λογισμικού και τον προγραμματισμό ανίχνευσης.**

Προσδοκώμενα Αποτελέσματα **Όταν θα έχετε μελετήσει την ενότητα, θα είστε σε θέση:**

<u>να αναφέρετε τεχνικές για ανίχνευση και εξάλειψη εξαρτήσεων,</u>

<u>να περιγράφετε τη σωλήνωση λογισμικού-συμβολική ανάπτυξη του βρόχου,</u>

<u>να παρουσιάζετε τις διαδικασίες απαιτούνται για την υλοποίηση του προγραμματισμού ανίχνευσης.</u>

σωλήνωση λογισμικού, συμβολική ανάπτυξη βρόχου, προγραμματισμός ανίχνευσης, επιλογή ανίχνευσης, συμπίεση ανίχνευσης

Ανίχνευση και εξάλειψη εξαρτήσεων

Η εύρεση των εξαρτήσεων ενός προγράμματος είναι ένα σημαντικό κομμάτι το οποίο απαιτεί την εκτέλεση τριών εργασιών**:**

1. Καλός προγραμματισμός του κώδικα.
2. Καθορισμός των βρόχων που μπορεί να περιέχουν παραλληλισμό.
3. Εξάλειψη των εξαρτήσεων ονόματος.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 1

Θυμάστε πότε προκύπτουν οι εξαρτήσεις ονομάτων και τα είδη στα οποία χωρίζονται; Για περισσότερες λεπτομέρειες καλό θα ήταν να ανατρέξετε στην δεύτερη ενότητα αυτού του κεφαλαίου: «Είδη εξαρτήσεων-Εξαρτήσεις δεδομένων και ονομάτων».

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 1

Η εξάρτηση ονομάτων προκύπτει όταν δύο εντολές χρησιμοποιούν τον ίδιο καταχωρητή ή την ίδια θέση μνήμης, που καλείται όνομα, αλλά δεν υπάρχει καμιά ροή δεδομένων μεταξύ των εντολών, που να σχετίζεται με το όνομα.
Υπάρχουν δύο είδη εξαρτήσεων ονομάτων μεταξύ μιας εντολής i, η οποία προηγείται της εντολής j:

1. **Η αντι-εξάρτηση μεταξύ της εντολής i και της εντολής j, προκύπτει όταν η εντολή j γράφει στον καταχωρητή ή στη θέση μνήμης, όπου η εντολή i διαβάζει και έχει εκτελεστεί πρώτη. Η αντι-εξάρτηση αντιστοιχεί σε κίνδυνο WAR και ο εντοπισμός των WAR κινδύνων, οδηγεί στη διάταξη ενός μη- εξαρτώμενου ζεύγους εντολών.**
2. **Η εξάρτηση εξόδου, προκύπτει όταν η εντολή i και η εντολή j γράφουν στον ίδιο καταχωρητή ή στην ίδια περιοχή μνήμης. Η διάταξη μεταξύ των εντολών πρέπει να διατηρείται. Οι εξαρτήσεις εξόδου διασφαλίζονται εντοπίζοντας WAW κινδύνους.**

Είναι αναγκαίο να βρούμε όλες τις εξαρτήσεις και να καθορίσουμε αν υπάρχει κάποιος κύκλος στις εξαρτήσεις, αφού το γεγονός αυτό μας εμποδίζει να τρέξουμε το βρόχο παράλληλα.

Παράδειγμα

Θεωρήστε το παρακάτω τμήμα κώδικα :

```
for  ( i = 1 ; i<= 100 ; i = i+1)
     {
      A[i] = B[i] + C[i]
      D[i] = A[i] * E[i]
     }
```

Επειδή η εξάρτηση που συνεπάγεται ο πίνακας Α δεν είναι εξάρτηση που μεταφέρεται με το βρόχο, μπορούμε να αναπτύξουμε το βρόχο και να εκτελέσουμε παράλληλα τις πολλαπλές επαναλήψεις του βρόχου, διατηρώντας όμως, τη σειρά για κάθε ζευγάρι εντολών σε κάθε επανάληψη. Η δεύτερη αναφορά στον πίνακα Α δε χρειάζεται να μεταφραστεί σε μια εντολή φόρτωσης, αφού η τιμή έχει ήδη υπολογιστεί από την προηγούμενη εντολή. Η βελτιστοποίηση αυτή, απαιτεί οι δύο αναφορές να είναι πάντα στην ίδια διεύθυνση μνήμης και ότι δεν μεσολαβεί πρόσβαση στην ίδια θέση.

Συνήθως, η ανάλυση της εξάρτησης δεδομένων δείχνει ότι μία αναφορά μπορεί να εξαρτάται από μία άλλη. Όμως, μία πιο πολύπλοκη ανάλυση απαιτείται για να καθορίσουμε ότι οι δύο αναφορές πρέπει να βρίσκονται ακριβώς στην ίδια διεύθυνση. Στο παραπάνω παράδειγμα, αρκεί μία απλή έκδοση αυτής της ανάλυσης, αφού οι αναφορές βρίσκονται στο ίδιο βασικό μπλοκ.

Σε πολλούς παράλληλους βρόχους ο βαθμός της παραλληλίας περιορίζεται μόνο από τον αριθμό των αναπτύξεων, ο οποίος περιορίζεται μόνο από τον αριθμό των επαναλήψεων του βρόχου. Βέβαια, στην πράξη, η εκμετάλλευση της παραλληλίας θα απαιτούσε πολλές λειτουργικές μονάδες και πιθανόν ένα τεράστιο αριθμό καταχωρητών. Η απουσία εξάρτησης που μεταφέρεται με το βρόχο, απλά δηλώνει ότι έχουμε διαθέσιμο ένα μεγάλο βαθμό παραλληλίας.

Συχνά οι εξαρτήσεις που μεταφέρονται με τον βρόχο εμφανίζονται στη μορφή μιας αναδρομής :

```
for  ( i = 2 ; i<= 100 ; i = i+1)
     {
      Y[i] = Y[i-1] + Y[i];
     }
```

Μία αναδρομή συμβαίνει όταν, μία μεταβλητή υπολογίζεται από την τιμή της από προηγούμενη επανάληψη και συχνά από την αμέσως προηγούμενη.
Η ανίχνευση των αναδρομών είναι σημαντική γιατί ένα πλήθος αναδρομών μπορεί να αποτελέσει την πηγή για μια λογική ποσότητα παραλληλίας. Αυτό φαίνεται από το παρακάτω παράδειγμα.

Παράδειγμα

Θεωρήστε τον παρακάτω βρόχο :

```
for  ( i = 6 ; i<= 100 ; i = i+1)
     {
      Y[i] = Y[i-5] + Y[i];
     }
```

Στην επανάληψη i, ο βρόχος αναφέρεται στο στοιχείο i-5. Λέμε ότι ο βρόχος έχει απόσταση εξάρτησης ίση με 5. Πολλοί βρόχοι με εξαρτήσεις που μεταφέρονται έχουν απόσταση εξάρτησης ίση με 1. Όσο μεγαλύτερη είναι η απόσταση, τόσο περισσότερος ενδεχόμενος παραλληλισμός μπορεί να επιτευχθεί με την ανάπτυξη του βρόχου. Για παράδειγμα, εάν αναπτύξουμε τον πρώτο βρόχο, με απόσταση εξάρτησης ίση με 1, τότε διαδοχικές εντολές αλληλοεξαρτώνται. Υπάρχει ακόμα παραλληλία ανάμεσα στις μεμονωμένες εντολές, αλλά όχι αρκετή. Εάν αναπτύξουμε τον βρόχο που έχει απόσταση εξάρτησης ίση με 5, υπάρχει μία ακολουθία πέντε εντολών που δεν έχουν εξαρτήσεις και έτσι υπάρχει περισσότερη παραλληλία σε επίπεδο εντολής. Αν και πολλοί βρόχοι με εξαρτήσεις μεταφερόμενες από τους βρόχους έχουν απόσταση εξάρτησης ίση με 1, προκύπτουν περιπτώσεις βρόχων με μεγαλύτερες αποστάσεις που παρέχουν αρκετή παραλληλία ώστε να κρατήσουν τον επεξεργαστή απασχολημένο.

Όλοι οι αλγόριθμοι ανάλυσης εξάρτησης δουλεύουν με την προϋπόθεση ότι οι δείκτες του πίνακα είναι συγγενικοί. Ο δείκτης ενός μονοδιάστατου πίνακα είναι συγγενικός όταν έχει τη μορφή a * i + b, όπου a και b σταθερές και i μεταβλητή δείκτη βρόχου.
Υποθέτουμε ότι αποθηκεύουμε σε ένα στοιχείο του πίνακα με δείκτη a * i + b και φορτώνουμε από τον ίδιο πίνακα με δείκτη c * i + d, όπου i είναι η μεταβλητή που παίρνει τιμή από m έως n. Μία εξάρτηση υπάρχει εάν ισχύουν δύο συνθήκες **:**

1. Υπάρχουν δύο δείκτες επανάληψης, j και k, με m<= j και k<= n.

2. Ο βρόχος αποθηκεύει σε ένα στοιχείο του πίνακα με δείκτη a * j + b και αργότερα ανακαλεί το ίδιο στοιχείο του πίνακα όταν αυτό έχει δείκτη c * k + d. Αυτό σημαίνει ότι a * j +b = c * k + d.

Μία απλή και επαρκής εξέταση για την απουσία εξάρτησης είναι ο μέγιστος κοινός διαιρέτης (GCD). Βασίζεται στην παρατήρηση ότι, εάν υπάρχει εξάρτηση μεταφερόμενη με τον βρόχο, τότε ο μέγιστος κοινός διαιρέτης των (c,a) πρέπει να διαιρεί το (d-b).

Παράδειγμα

Να χρησιμοποιήσετε το μέγιστο κοινό διαιρέτη για να καθορίσετε αν υπάρχουν εξαρτήσεις στον ακόλουθο βρόχο :

```
for  ( i = 1 ; i<= 100 ; i = i+1)
      {
      X[2*i+3] = X[2*i] * 5.0 ;
      }
```

Δίνοντας τις τιμές a=2, b=3, c=2 και d=0, τότε GCD (a,c)=2 και d-b= -3. Αφού το 2 δεν διαιρεί το –3, καμία εξάρτηση δεν είναι πιθανή.

Για να χρησιμοποιηθεί ο μέγιστος κοινός διαιρέτης, ο βρόχος θα πρέπει πρώτα να κανονικοποιήσουμε το βρόχο, δηλαδή να γραφτεί έτσι ώστε ο δείκτης να αρχίζει από ένα και να αυξάνεται κατά ένα σε κάθε επανάληψη.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 2

Να χρησιμοποιήστε το μέγιστο κοινό διαιρέτη για να καθορίσετε αν υπάρχουν εξαρτήσεις στον ακόλουθο βρόχο:

```
for   (i=2 ; i<=1000 ; i+=2)
      {
      a[i]=a[i+2];
      }
```

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 2

Αρχικά θα πρέπει να κανονικοποιήσουμε το βρόχο, δηλαδή να γραφτεί έτσι ώστε ο δείκτης να αρχίζει από ένα και να αυξάνεται κατά ένα σε κάθε επανάληψη.
Μία κανονικοποιημένη έκδοση του βρόχου είναι η ακόλουθη:

```
for (i=1 ; i<=500; i+=1)
    { a[2*i]=a[2*i+2]; }
```

Αν θεωρήσουμε a=2, b=0, c=2 και d=2, τότε έχουμε:
GCD (a,c) = GCD (2,2) = 2 και d-b = 2
Το 2 διαιρεί το 2 οπότε υπάρχουν εξαρτήσεις.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 3

Να χρησιμοποιήστε το μέγιστο κοινό διαιρέτη για να καθορίσετε αν υπάρχουν εξαρτήσεις στον ακόλουθο βρόχο:

```
for   (i=2 ; i<=100 ; i+=2)
      {
       a[i]=a[50*i+1];
      }
```

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 3

Αρχικά θα πρέπει να κανονικοποιήσουμε το βρόχο, δηλαδή να γραφτεί έτσι ώστε ο δείκτης να αρχίζει από ένα και να αυξάνεται κατά ένα σε κάθε επανάληψη.
Μία κανονικοποιημένη έκδοση του βρόχου είναι η ακόλουθη:

```
for   (i=1 ; i<=50 ; i++)
      { a[2*i]=a[100*i+1]; }
```

Σύμφωνα με τη θεωρία οι τιμές των a, b, c και d είναι:
a=2, b=0, c=100 και d=1.
Ο μέγιστος κοινός διαιρέτης των a και c είναι GCD (a,c) = GCD (2,100) = 2 και d-b = 1-0=1.
Παρατηρούμε ότι το 2 δεν διαιρεί το 1, άρα δεν υπάρχει κάποια εξάρτηση.

Υπάρχουν περιπτώσεις όπου ο μέγιστος κοινός διαιρέτης αν και είναι επιτυχής δεν υπάρχουν εξαρτήσεις. Αυτό συμβαίνει, για παράδειγμα, όταν ο μέγιστος κοινός διαιρέτης δεν λαμβάνει υπόψη του τα όρια του βρόχου.

Παράδειγμα

Θεωρήστε τον παρακάτω βρόχο :

```
for  ( i = 1 ; i<= 100 ; i = i+1)
     {
     Y[i] = X[i] / c ;      /*S1*/
     X[i] = X[i] + c;       /*S2*/
     Z[i] = Y[i] + c;       /*S3*/
     Y[i] = c - Y[i] ;      /*S4*/
     }
```

Να βρείτε όλες τις πραγματικές εξαρτήσεις, τις εξαρτήσεις εξόδου και τις αντι-εξαρτήσεις. Στη συνέχεια να εξαλείψετε τις εξαρτήσεις εξόδου και τις αντι-εξαρτήσεις με χρήση της μεθόδου της μετονομασίας.

Απάντηση:

Ανάμεσα στις τέσσερις εντολές του βρόχου υπάρχουν οι ακόλουθες εξαρτήσεις **:**

1. Υπάρχουν πραγματικές εξαρτήσεις μεταξύ των εντολών S1 και S3 και μεταξύ των S1 και S4, λόγω του Y[i]. Αυτές δεν είναι εξαρτήσεις που μεταφέρονται με τον βρόχο και επομένως πολλαπλές επαναλήψεις του βρόχου μπορούν να εκτελεστούν παράλληλα. Οι εξαρτήσεις αυτές αναγκάζουν τις εντολές S3 και S4 να περιμένουν την ολοκλήρωση της εντολής S1.
2. Υπάρχει μία αντι-εξάρτηση μεταξύ των εντολών S1 και S2 λόγω του X[i].
3. Υπάρχει μία αντι-εξάρτηση μεταξύ των εντολών S3 και S4 λόγω του Y[i].
4. Υπάρχει μία εξάρτηση εξόδου μεταξύ των εντολών S1 και S4 λόγω του Y[i].

Η παρακάτω εκδοχή του βρόχου εξαλείφει τις εξαρτήσεις αυτές **:**

```
for  ( i = 1 ; i<= 100 ; i = i+1)
     {
     T[i]  = X[i] / c ;      /*S1*/
     X1[i] = X[i] + c;       /*S2*/
     Z[i]  = T[i] + c;       /*S3*/
     Y[i]  = c - T[i] ;      /*S4*/
     }
```

Στην εντολή S1 ο Y μετονομάστηκε σε T με σκοπό τη μετακίνηση της εξάρτησης εξόδου. Στην εντολή S2 ο X μετονομάστηκε σε X1 με σκοπό τη μετακίνηση της αντι-εξάρτησης. Στις εντολές S3 και S4 ο Y μετονομάστηκε σε T με σκοπό τη μετακίνηση της αντι-εξάρτησης.

Σωλήνωση λογισμικού-Συμβολική ανάπτυξη του βρόχου

Γνωρίζουμε ότι μία τεχνική του μεταγλωττιστή, για αποκάλυψη περισσότερης παραλληλίας ανάμεσα στις εντολές, είναι η ανάπτυξη του βρόχου. Υπάρχουν άλλες δύο σημαντικές τεχνικές που αναπτύχθηκαν γι' αυτό το σκοπό. Αυτές είναι **:**

1. Σωλήνωση λογισμικού.
2. Προγραμματισμός ανίχνευσης.

Η σωλήνωση λογισμικού αναδιοργανώνει τους βρόχους έτσι ώστε, κάθε επανάληψη του σωληνωμένου λογισμικού κώδικα να σχηματίζεται από εντολές επιλεγμένες, από διαφορετικές επαναλήψεις του αρχικού βρόχου.

Παράδειγμα

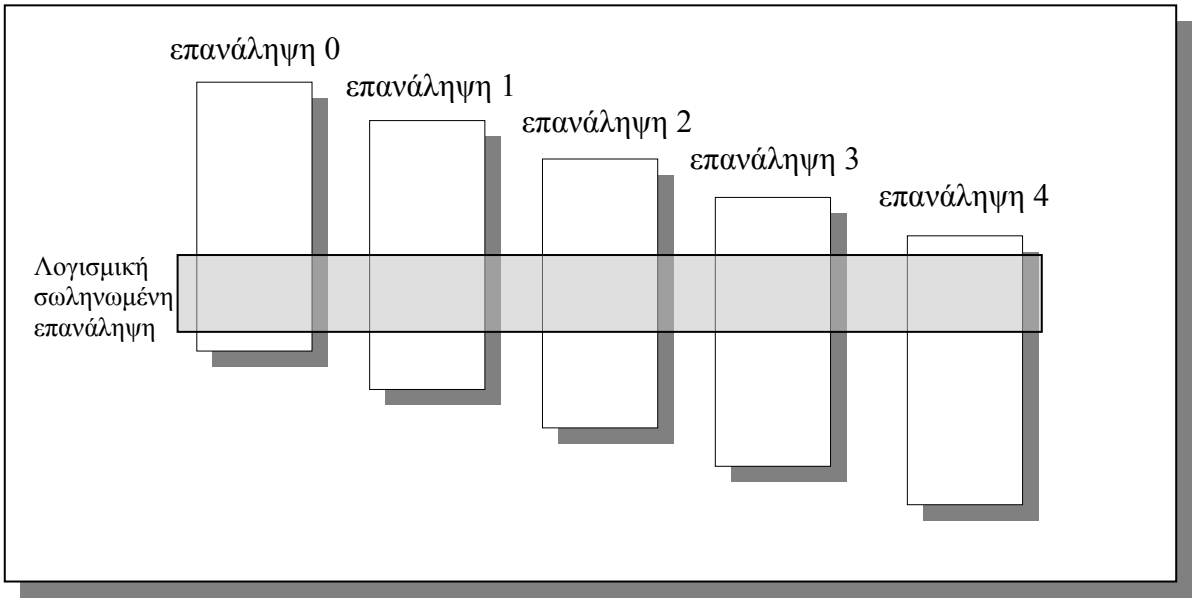
Για τον παρακάτω βρόχο δίνεται ο αντίστοιχος πίνακας με τις επιλεγμένες εντολές από τις διαφορετικές επαναλήψεις του βρόχου **:**

```
loop: LD      F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD    F4, F0, F2  ;   πρόσθεση με το βαθμωτό μέγεθος
                                του F2
      SD      0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI    R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ    R1, loop    ;   διακλάδωση R1!= μηδέν
```

Εντολή ακεραίου	FP εντολή	Κύκλοι ρολογιού
loop: LD F0, 0 (R1)		1
LD F6, -8 (R1)		2
LD F10, -16 (R1)	ADDD F4, F0, F2	3
LD F14, -24 (R1)	ADDD F8, F6, F2	4
LD F18, -32 (R1)	ADDD F12, F10, F2	5
SD 0 (R1), F4	ADDD F16, F14, F2	6
SD -8 (R1), F8	ADDD F20, F18, F2	7
SD -16 (R1), F12		8
SD -24 (R1), F16		9
SUBI R1, R1, #40		10
BNEZ R1, loop		11
SD -32 (R1), F20		12

Στον πίνακα έχουμε πέντε αντίγραφα των εντολών LD, ADDD και SD από διαφορετικές επαναλήψεις του βρόχου και ένα αντίγραφο των εντολών SUBI και BNEZ.

Η λογισμική σωλήνωση του βρόχου επιλέγει εντολές από διαφορετικές επαναλήψεις του βρόχου, χωρίς να χρειάζεται η ανάπτυξή του, διαχωρίζοντας έτσι τις εξαρτώμενες εντολές που υπάρχουν μέσα σε μια επανάληψη του βρόχου. Σχηματικά παρουσιάζεται στο παρακάτω σχήμα.



Παράδειγμα

Δίνεται ο παρακάτω βρόχος :

```
loop: LD      F0, 0 (R1)  ;   F0 = στοιχείο πίνακα
      ADDD    F4, F0, F2  ;   πρόσθεση με το βαθμωτό μέγεθος
                              του F2
      SD      0 (R1), F4  ;   αποθήκευση του αποτελέσματος
      SUBI    R1, R1, #8  ;   μείωση δείκτη 8 bytes (για κάθε DW)
      BNEZ    R1, loop    ;   διακλάδωση R1!= μηδέν
```

Να δείξετε μία έκδοση του βρόχου με σωλήνωση λογισμικού, η οποία αυξάνει όλα τα στοιχεία ενός πίνακα, του οποίου η αρχική διεύθυνση είναι η R1, από τα περιεχόμενα του F2.

Απάντηση:

Η σωλήνωση λογισμικού συμβολικά αναπτύσσει το βρόχο και μετά επιλέγει εντολές από κάθε επανάληψη. Μια και η ανάπτυξη του βρόχου είναι συμβολική, οι εντολές SUBI και BNEZ δεν χρειάζεται να εμφανιστούν σε αυτό το στάδιο. Παρακάτω παρουσιάζονται οι εντολές τριών διαδοχικών επαναλήψεων **:**

```
Επανάληψη i :       LD      F0, 0 (R1)
                    ADDD    F4, F0, F2
                    SD      16 (R1), F4
Επανάληψη i + 1 :   LD      F0, 0 (R1)
                    ADDD    F4, F0, F2
                    SD      16 (R1), F4
Επανάληψη i + 2 :   LD      F0, 0 (R1)
                    ADDD    F4, F0, F2
                    SD      16 (R1), F4
```

Μετά, επιλέγουμε εντολές από κάθε επανάληψη και τις τοποθετούμε στο βρόχο μαζί με τις εντολές ελέγχου, SUBI και BNEZ. Έτσι έχω **:**

```
loop: SD      0 (R1), F4   ;  αποθηκεύει στο M[i]
      ADDD    F4, F0, F2   ; προσθέτει στο M[i-1]
      LD      F0, 0 (R1)   ; φορτώνει το M[i-2]
      SUBI    R1, R1, #8
      BNEZ    R1, loop
```

Ο βρόχος αυτός μπορεί να τρέξει σε πέντε κύκλους ανά αποτέλεσμα. Επειδή η εντολή φόρτωσης χωρίζεται από την εντολή αποθήκευσης κατά 16, ο βρόχος θα τρέχει με δύο λιγότερες επαναλήψεις.

Η σωλήνωση λογισμικού μπορεί να θεωρηθεί ως η συμβολική ανάπτυξη του βρόχου.

ΔΡΑΣΤΗΡΙΟΤΗΤΑ 4

Μπορείτε να προσδιορίσετε ποιο είναι το πλεονέκτημα της σωλήνωσης λογισμικού;

ΑΠΑΝΤΗΣΗ ΔΡΑΣΤΗΡΙΟΤΗΤΑΣ 4

Το πλεονέκτημα της σωλήνωσης λογισμικού είναι ότι καταναλώνει λιγότερο κώδικα.

Προγραμματισμός ανίχνευσης

Ο Προγραμματισμός ανίχνευσης επεκτείνει την ανάπτυξη του βρόχου με μία τεχνική για εύρεση παραλληλισμού διαμέσου διακλαδώσεων υπό συνθήκη, εκτός από τις διακλαδώσεις του βρόχου.

Ο Προγραμματισμός ανίχνευσης είναι χρήσιμος σε επεξεργαστές με πολύ μεγάλο αριθμό διευθετήσεων ανά κύκλο ρολογιού.

Ο Προγραμματισμός ανίχνευσης είναι ένας συνδυασμός δύο ανεξάρτητων διαδικασιών **:**

- Η πρώτη διαδικασία ονομάζεται ***επιλογή ανίχνευσης*** και προσπαθεί να βρει μία πιθανή ακολουθία βασικών μπλοκ, των οποίων οι λειτουργίες θα τοποθετηθούν μαζί σε ένα μικρότερο αριθμό εντολών. Η ακολουθία αυτή ονομάζεται ***ανίχνευση***.

- Η δεύτερη διαδικασία ονομάζεται ***συμπίεση ανίχνευσης*** και προσπαθεί να συμπιέσει την επιλεγόμενη ανίχνευση σε ένα μικρό αριθμό μεγάλων εντολών, διατηρώντας τις εξαρτήσεις δεδομένων και ελέγχου.

Το πλεονέκτημα του προγραμματισμού ανίχνευσης είναι η παροχή ενός σχήματος, το οποίο μειώνει τις συνέπειες των εξαρτήσεων ελέγχου, μετακινώντας τον κώδικα διαμέσου διακλαδώσεων υπό συνθήκη, που δεν ανήκουν σε βρόχο, χρησιμοποιώντας την πρόβλεψη διακλάδωσης.

Η ανάπτυξη του βρόχου, η σωλήνωση λογισμικού και ο προγραμματισμός ανίχνευσης στοχεύουν στην αύξηση της παραλληλίας, η οποία μπορεί να αξιοποιηθεί από έναν επεξεργαστή που διευθετεί περισσότερες από μία εντολές σε κάθε κύκλο ρολογιού.

Ανακεφαλαιώνοντας λοιπόν…

Η εύρεση των εξαρτήσεων ενός προγράμματος είναι ένα σημαντικό κομμάτι το οποίο απαιτεί την εκτέλεση τριών εργασιών :

1. **Καλός προγραμματισμός του κώδικα.**
2. **Καθορισμός των βρόχων που μπορεί να περιέχουν παραλληλισμό.**
3. **Εξάλειψη των εξαρτήσεων ονόματος.**

Σε πολλούς παράλληλους βρόχους ο βαθμός της παραλληλίας περιορίζεται μόνο από τον αριθμό των αναπτύξεων, ο οποίος περιορίζεται μόνο από τον αριθμό των επαναλήψεων του βρόχου.

Μία απλή και επαρκής εξέταση για την απουσία εξάρτησης είναι ο μέγιστος κοινός διαιρέτης (GCD).

Η σωλήνωση λογισμικού αναδιοργανώνει τους βρόχους έτσι ώστε, κάθε επανάληψη του σωληνωμένου λογισμικού κώδικα να σχηματίζεται από εντολές επιλεγμένες, από διαφορετικές επαναλήψεις του αρχικού βρόχου.

Ο Προγραμματισμός ανίχνευσης επεκτείνει την ανάπτυξη του βρόχου με μία τεχνική για εύρεση παραλληλισμού διαμέσου διακλαδώσεων υπό συνθήκη, εκτός από τις διακλαδώσεις του βρόχου.