



Παράλληλη Επεξεργασία

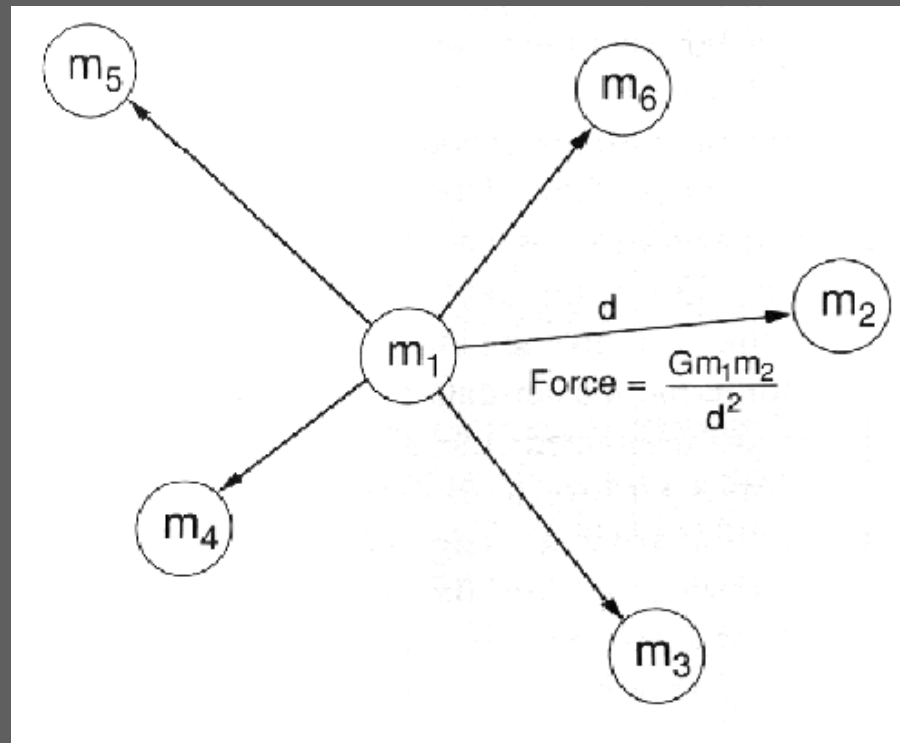
Κεφάλαιο 9^ο

Επιμερισμός Δεδομένων

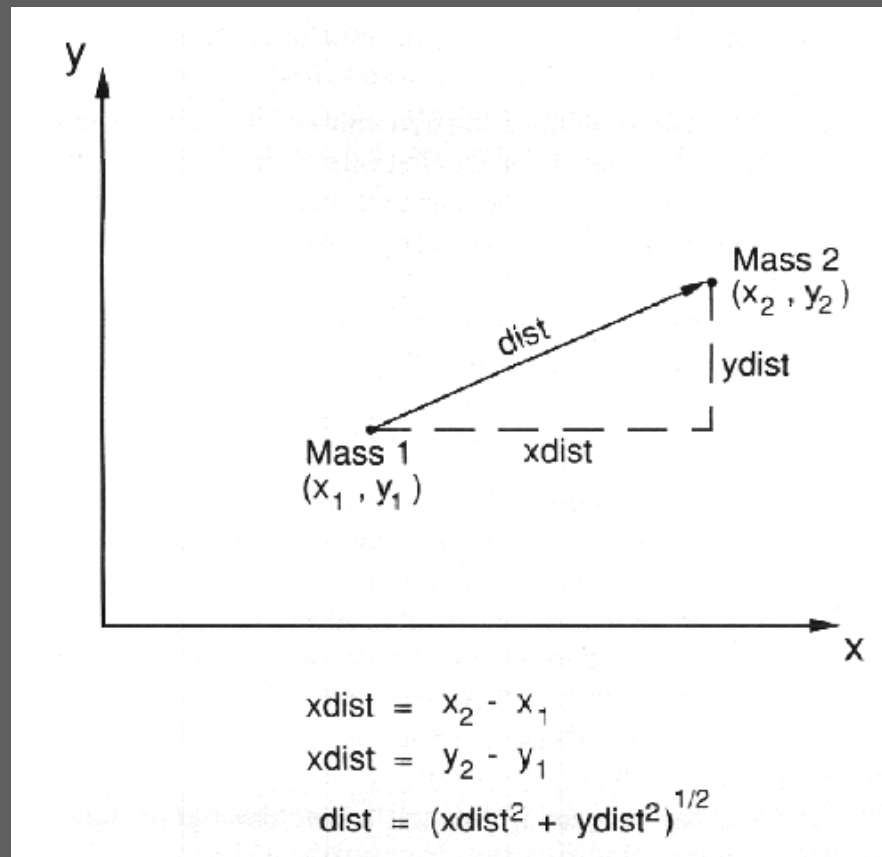
Κωνσταντίνος Μαργαρίτης
Καθηγητής
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
kmarg@uom.gr
<http://eos.uom.gr/~kmarg>

Αρετή Καπτάν
Υποψήφια Διδάκτορας
Τμήμα Εφαρμοσμένης Πληροφορικής
Πανεπιστήμιο Μακεδονίας
areti@uom.gr
<http://eos.uom.gr/~areti>

Πρόβλημα N σωμάτων στην Αστροφυσική



Υπολογισμός Δυνάμεων Βαρύτητας



Πρόβλημα N σωμάτων σε Παράλληλο Σύστημα Διαμοιραζόμενης Μνήμης

```
PROGRAM Nbody;
CONST      n=100;          (*Αριθμός των σωμάτων*)
           G=...;         (*Σταθερά βαρύτητας*)
TYPE
bodytype=RECORD
  mass, x, y: REAL; (*Μάζα και θέση*)
END;
forcetype=RECORD
  x, y: REAL; (*Δύναμη στις κατευθύνσεις x και y*)
END;
VAR
  body: ARRAY [1..n] OF bodytype; (*Αρχικός πίνακας δεδομένων*)
  force: ARRAY [1..n] OF forcetype; (*προκύπτουσες δυνάμεις*)
  i: INTEGER;

PROCEDURE FindForce(me: INTEGER);
VAR
  j: INTEGER;
  xdist, ydist, dist, distsq, pull: REAL;
BEGIN
  force[me].x:= 0; force[me].y:= 0;
  FOR j:= 1 TO n DO (*Εξέταση κάθε άλλου σώματος*)
    IF j <> me DO BEGIN
      xdist:= body[j].x - body[me].x;
      ydist:= body[j].y - body[me].y;
      distsq:= xdist*xdist + ydist*ydist;
      dist:= Sqrt(distsq); (*Απόσταση μεταξύ των σωμάτων*)
      pull:= G*body[me].mass * body[j].mass/distsq;
      force[me].x:= force[me].x + pull*xdist/dist; (*Η δύναμη x*)
      force[me].y:= force[me].y + pull*ydist/dist; (*Η δύναμη y*)
    END;
  END;
END;
```

Πρόβλημα N σωμάτων σε Παράλληλο Σύστημα Κατανεμημένης Μνήμης

```
For each body  $i$  in my “permanenet partition” do
    For each body  $j$  in the “circulating partition” do
        Compute force exerted by body  $j$  on body  $i$ ;
Send circulating partition to the right-neighbor process;
Receive new circulating partition from left-neighbor process;
```

Πρόβλημα N Σωμάτων σε Δακτύλιο

25 επεξεργασιών

```
PROGRAM Nbody;
ARCHITECTURE RING(25);
CONST      n=100; (*Αριθμός των σωμάτων*)
           numproc=25; (*Αριθμός των διεργασιών*)
           partsize=4; (*Μέγεθος κάθε τμήματος*)
           G=...;      (*Σταθερά βαρύτητας*)
TYPE      bodytype=    RECORD
                m, x, y: REAL; (*Μάζα και θέση*)
           END;
           forcetype=  RECORD;
                x, y:REAL; (*Δύναμη στις κατευθύνσεις x και y*)
           END;
           parttype=   ARRAY [1..partsize] OF bodytype;
           resulttype=ARRAY [1..partsize] OF forcetype;
VAR      bodies: ARRAY [1..numproc] OF parttype; (*Αρχικός πίνακας δεδομένων*)
           finalforce: ARRAY [1..numproc] OF resulttype; (*Προκύπτουσες δυνάμεις*)
           i, j: INTEGER;
           inchan: ARRAY [1..numproc] OF CHANNEL OF parttype;      (*Επικοινωνία*)

PROCEDURE FindForce(me: INTEGER; body: parttype; (*Το μόνιμο τμήμα*)
                   VAR result: resulttype); (*Για να σταλούν πίσω τα αποτελέσματα*)
VAR      i, j, k: INTEGER;
           circ: parttype; (*Περιστρεφόμενο τμήμα*)
           force: resulttype; (*Για την συσσωρευμένη δύναμη στα σώματα*)
           xdist, ydist, dist, sqdist, pull: REAL;
```

(...Συνέχεια...)

Πρόβλημα N Σωμάτων σε Δακτύλιο 25 επεξεργαστών

BEGIN

circ:= body; (*Αντιγραφή του μόνιμου τμήματος στο περιστρεφόμενο*)

FOR i:= 1 TO partsize DO BEGIN

force[i].x:= 0; force[i].y:= 0; END;

FOR k:= 1 TO numproc DO BEGIN

FOR i:= 1 TO partsize DO

FOR j:= 1 TO partsize DO

BEGIN (*Υπολογισμός της δύναμης που ασκείται από το σώμα j
στο σώμα i*)

xdist:= circ[j].x - body[i].x;

ydist:= circ[j].y - body[i].y;

distsq:= xdist*xdist + ydist*ydist;

dist:= Sqrt(distsq);

IF dist <> 0 THEN BEGIN

pull:= G * body[i].m * circ[j].m / distsq;

force[i].x:= force[i].x + pull*xdist/dist;

force[i].y:= force[i].y + pull*ydist/dist;

END;

END;

inchan[me MOD numproc + 1]:= circ; (*Αποστολή στα δεξιά*)

circ:= inchan[me]; (*Λήψη από τον αριστερό γείτονα*)

END;

result:= force; (*Τελική απάντηση στην πρωταρχική διεργασία*)

END;

BEGIN (* Κυρίως πρόγραμμα*)

.... (*Αρχικοποίηση του πρωταρχικού πίνακα "bodies"*)

FORALL i:= 1 TO numproc DO (*Δημιουργία των διεργασιών*)

(@i-1 PORT inchan[i]) FindForce(i, bodies[i], finalforce[i]);

END.

Ακολουθία δραστηριοτήτων κατά τη διάρκεια κάθε επανάληψης

- ⇒ Υπολογισμός της δύναμης που ασκείται από όλα τα σώματα του περιστρεφόμενου τμήματος σε όλα τα σώματα του μόνιμου τμήματος.
- ⇒ Αποστολή του τρέχοντος περιστρεφόμενου τμήματος στο δεξιό γείτονα.
- ⇒ Λήψη του νέου περιστρεφόμενου τμήματος από τον αριστερό γείτονα.

Επικάλυψη χρόνου επικοινωνίας/υπολογισμού

```
PROCEDURE FindForce(me: INTEGER;body: parttype; (*Το μόνιμο τμήμα*)
    VAR result: resulttype); (*Για την αποστολή των αποτελεσμάτων πίσω*)
VAR
    i, j, k: INTEGER;
    circ: parttype; (*Περιστρεφόμενο τμήμα*)
    force: resulttype; (*Για τη συσσώρευση της δύναμης στα σώματα*)
    xdist, ydist, dist, distsq, pull: REAL;

BEGIN
    circ:= body; (*Αντιγραφή του μόνιμου τμήματος στο περιστρεφόμενο*)
    FOR i:= 1 TO partsize DO BEGIN force[i].x:= 0; force[i].y:= 0; END;
    FOR k:= 1 TO numproc DO BEGIN
        inchan[me MOD numproc+1]:= circ; (*Αποστολή στα δεξιά**)
        FOR i:= 1 TO partsize DO (*Φάση Υπολογισμού*)
            FOR j:= 1 TO partsize DO BEGIN
                (*Υπολογισμός της δύναμης που ασκείται από το σώμα j στο σώμα i*)
                xdist:= circ[j].x - body[i].x;
                ydist:= circ[j].y - body[i].y;
                distsq:= xdist*xdist + ydist*ydist;
                dist:= Sqrt(distsq);
                IF dist <> 0 THEN BEGIN
                    pull:= G * body[i].m * circ[j].m/distsq;
                    force[i].x:= force[i].x + pull*xdist/dist;
                    force[i].y:= force[i].y + pull*ydist/dist;
                END;
            END;
        circ:= inchan[me]; (*Λήψη από τον αριστερό γείτονα*)
    END;
    result:= force; (*Τελική απάντηση πίσω στην πρωταρχική διεργασία*)
END;
```

Πλεονεκτήματα Τοπολογικής Απεικόνισης

- ⇒ Ιδεατή έναντι φυσικής τοπολογίας
- ⇒ Μεγάλη συνδεσιμότητα \Rightarrow Μικρότερη διάμετρος
- ⇒ Μείωση των χρόνων για:
 - Δημιουργία,
 - αρχικοποίηση και
 - τερματισμό των διεργασιών

Τοπολογίες με χαμηλή συνδεσιμότητα μπορούν να απεικονιστούν σε τοπολογίες με υψηλή συνδεσιμότητα

Τοπολογίες κατά αύξουσα συνδεσιμότητα

- ⇒ Γραμμή
- ⇒ Δακτύλιος
- ⇒ Πλέγμα δύο διαστάσεων
- ⇒ Τόρος
- ⇒ Πλέγμα τριών διαστάσεων
- ⇒ Υπερκύβος
- ⇒ Πλήρως Συνδεδεμένη

Ακολουθία κώδικα Gray 3-bits.

Gray[1]=0 (δυαδικό 000)

Gray[2]=1 (δυαδικό 001)

Gray[3]=3 (δυαδικό 011)

Gray[4]=2 (δυαδικό 010)

Gray[5]=6 (δυαδικό 110)

Gray[6]=7 (δυαδικό 111)

Gray[7]=5 (δυαδικό 101)

Gray[8]=4 (δυαδικό 100)

Απεικόνιση ιδεατής τοπολογίας σε φυσική τοπολογία

⇒ Τοπολογία δακτυλίου

```
FORALL i:= 1 TO numproc DO (*Δημιουργία των διεργασιών*)  
    (@i-1 PORT inchan[i]) Findforce(i, bodies[i], finalforce[i]);
```

⇒ Τοπολογία υπερκύβου

```
FORALL i:= 1 TO numproc DO (*Δημιουργία των διεργασιών*)  
    (@Gray[i] PORT inchan[i]) FindForce(i, bodies[i], finalforce[i]);
```

Επιμερισμός για τον Πολλαπλασιασμό Μητρών

A	x	B	=	C																																																																											
<table border="1"><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td>A_{41}</td><td>A_{42}</td><td>A_{43}</td><td>A_{44}</td><td>A_{45}</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>																A_{41}	A_{42}	A_{43}	A_{44}	A_{45}							<table border="1"><tr><td></td><td>B_{12}</td><td></td><td></td><td></td></tr><tr><td></td><td>B_{22}</td><td></td><td></td><td></td></tr><tr><td></td><td>B_{32}</td><td></td><td></td><td></td></tr><tr><td></td><td>B_{42}</td><td></td><td></td><td></td></tr><tr><td></td><td>B_{52}</td><td></td><td></td><td></td></tr></table>		B_{12}					B_{22}					B_{32}					B_{42}					B_{52}					<table border="1"><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>C_{42}</td><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>																	C_{42}								
A_{41}	A_{42}	A_{43}	A_{44}	A_{45}																																																																											
	B_{12}																																																																														
	B_{22}																																																																														
	B_{32}																																																																														
	B_{42}																																																																														
	B_{52}																																																																														
	C_{42}																																																																														

Πολλαπλασιασμός Μητρώων σε Τοπολογία Τόρου

```
VAR    myA, myB, myC: REAL
        i: INTEGER;
BEGIN
    myC:= 0;
    FOR i:= 1 TO 3 DO BEGIN
        myC:= myC + myA*myB;
        send myA to neighbor processor in leftward
rotation;
        send myB to neighbor processor in upward
rotation;
        receive new myA;
        receive new myB;
    END;
    write myC to master product array;
END;
```

Απλοποιημένος Πολλαπλασιασμός Μητρών

```
PROGRAM Matrixmult;
ARCHITECTURE TORUS(3);
CONST n=3;
VAR      A, B, C: ARRAY [0..n-1, 0..n-1] OF REAL; (*Αρχικοί πίνακες δεδομένων*)
        Achan, Bchan: ARRAY [0..n-1, 0..n-1] OF CHANNEL OF REAL;
        i, j: INTEGER;

PROCEDURE Multiply(row, col: INTEGER; myA, myB: REAL; VAR mainC: REAL);
VAR      iter, above, left: INTEGER;
        myC: REAL;

BEGIN
    IF row > 0 THEN above:= row-1 ELSE above:= n-1; (*Πάνω γείτονας*)
    IF col > 0 THEN left:= col-1 ELSE left:=n-1; (*Αριστερός γείτονας*)
    myC:= 0;
    FOR iter:= 1 TO n DO BEGIN
        Achan[row, left]:= myA; (*Αποστολή του myA στην αριστερή περιστροφή*)
        Bchan[above, col]:= myB; (*Αποστολή του myB στην ανιούσα περιστροφή*)
        myC:= myC + myA*myB;
        myA:= Achan[row, col]; (*Λήψη του νέου myA*)
        myB:= Bchan[above, col]; (*Λήψη του νέου myB*)
    END;
    mainC:= myC; (*Αποστολή της τελικής τιμής στην κύρια διεργασία*)
END;

END;

BEGIN
    ... (*Αρχικοποίηση των τιμών των μητρών A και B*)
    FOR i:= 0 TO n-1 DO
        FOR j:= 0 TO n-1 DO FORK(@i*n+j PORT Achan[i,j]; Bchan[i,j]) Multiply(i, j, A[i, (j+i) MOD n], B[(i+j) MOD n, j], C[i,j]);
    END.
END.
```


Πολλαπλασιασμός Μητρώων σε Τόρο (M.I)

```
PROGRAM Matrixmult;
ARCHITECTURE TORUS(7);
CONST      m= 7; (*Ο Τόρος έχει mxm επεξεργαστές*)
           p= 5; (*Το μέγεθος του τμήματος είναι pxp*)
TYPE      partition= ARRAY [1..p, 1..p] OF REAL;
           chantype= CHANNEL OF partition;
VAR       A, B, C: ARRAY [0..m-1, 0..m-1] OF partition; (*Αρχικοί πίνακες δεδομένων*)
           Achan, Bchan: ARRAY [0..m-1, 0..m-1] OF chantype; (*Επικοινωνία*)
           i, j: INTEGER;

PROCEDURE Multiply(row,col:INTEGER; myA,myB: partition; VAR mainC: partition);
VAR       i, j, k, iter, above, left: INTEGER;
           myC: partition;
BEGIN
  IF row > 0 THEN above:= row-1 ELSE above:= m-1;
  IF col > 0 THEN left:= col-1 ELSE left:= m-1;
  FOR i:= 1 TO p DO
    FOR j:= 1 TO p DO
      myC[i,j]:= 0;
  FOR iter:= 1 TO m DO BEGIN
    Achan[row,left]:= myA; (*Αποστολή του myA στην αριστερή περιστροφή*)
    Bchan[above,col]:= myB; (*Αποστολή του myB στην ανιούσα περιστροφή*)
    FOR i:= 1 TO p DO (*Πολλαπλασιασμός των A και B τμημάτων*)
      FOR j:= 1 TO p DO
        FOR k:= 1 TO p DO
          myC[i,j]:= myC[i,j]+myA[i,k]*myB[k,j];
          myA:= Achan[row,col] (*Λήψη του νέου myA*)
          myB:= Bchan[above,col] (*Λήψη του νέου myB*)
```

```
END;
```

Πολλαπλασιασμός Μητρών σε Τόρο (M.II)

```
mainC:= myC; (*Εγγραφή του αποτελέσματος πίσω στην αρχική C*)  
END;
```

```
BEGIN
```

```
... (*Αρχικοποίηση των τιμών για τις μήτρες A και B*)
```

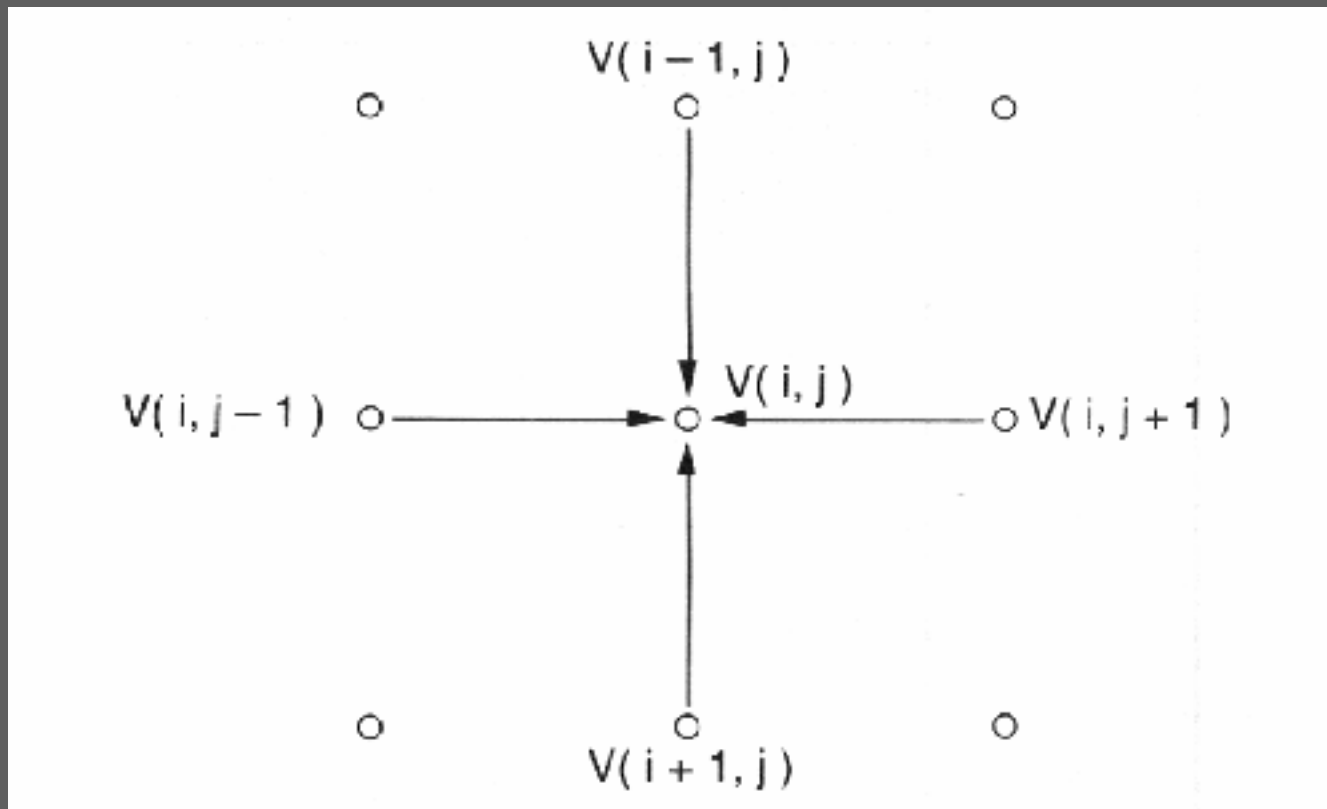
```
FOR i:= 0 TO m-1 DO
```

```
  FOR j:= 0 TO m-1 DO
```

```
    FORK(@i*m+j PORT Achan[i,j];Bchan[i,j])  Multiply(i, j, A[i, (j+i) MOD m], B[(i+j) MOD m, j], C[i,j]);
```

```
END.
```

Υπολογισμός της τιμής κάθε σημείου – Αλγόριθμος Jacobi



Ακολουθιακός αλγόριθμος του Jacobi

```
PROGRAM Jacobi;
CONST      n=32;
           tolerance= 0.1;
VAR        A, B: ARRAY [0..n+1,0..n+1] OF REAL;
           i,j : INTEGER;
           change,maxchange: REAL;

BEGIN
  ... (*Ανάγνωση των αρχικών τιμών για τον πίνακα A*)
  B:= A;
  REPEAT (*Υπολογισμός των νέων τιμών μέχρι να επιτευχθεί η επιθυμητή ανοχή*)
    BEGIN
      maxchange:= 0;
      FOR i:= 1 TO n DO
        FOR j:=1 TO n DO BEGIN (*Υπολογισμός της νέας τιμής και της αλλαγής της σε σχέση
                               με την παλιά τιμή*)
          B[i,j]:= (A[i-1,j]+A[i+1,j]+A[i,j-1]+A[i,j+1])/4;
          change:= ABS(B[i,j]-A[i,j]);
          If change > maxchange THEN maxchange:= change;
        END;
      A:= B;
    END;
  UNTIL maxchange < tolerance;
END.
```

Παράλληλος αλγόριθμος του Jacobi

```
VAR myrow,downrow,uprow,newrow: ARRAY [0..n+1] OF REAL;  
    j: INTEGER; done: BOOLEAN;  
    maxchange: REAL;  
  
BEGIN  
    REPEAT  
        FOR j:= 1 TO n DO (*Μέσος όρος των τεσσάρων γειτονικών σημείων*)  
            newrow[j]:= (myrow[j-1]+myrow[j+1]+downrow[j]+uprow[j])/4;  
            myrow:= newrow;  
            Send myrow to neighbors above and below;  
            Receive new copies of “downrow” and “uprow” from neighbors;  
            Compute “maxchange” in my row;  
            done:= Aggregate(maxchange<tolerance); (*Έλεγχος τερματισμού*)  
        UNTIL done;  
    END;
```

Jacobi Relaxation (M.I)

```
PROGRAM Jacobi;
ARCHITECTURE HYPERCUBE(5);
CONST      n=32; (*Αριθμός των επεξεργαστών*)
           d=5; (*Διάσταση του Υπερκύβου*)
           numiter=2*d; (*Πλήθος των επαναλήψεων πριν τον έλεγχο τερματισμού*)
           tolerance=.1;

TYPE      rowtype= ARRAY [0..n+1] OF REAL;
VAR       A: ARRAY [0..n+1] OF rowtype;
           i: INTEGER;
           upchan,downchan: ARRAY [1..n] OF CHANNEL OF rowtype; (*Θύρες επικοινωνίας*)
           GrayCode: ARRAY [1..n] OF INTEGER;
           inchan: ARRAY [0..n-1,1..d] OF CHANNEL OF BOOLEAN; (*Για την Συλλογή*)

FUNCTION Aggregate(mydone: BOOLEAN): BOOLEAN;
....(*Συνάρτηση Πολλαπλής Συλλογής από το σχήμα 8.11*)

PROCEDURE Updaterow(me: INTEGER; myrow: rowtype; VAR out: rowtype);
VAR       j,k: INTEGER; maxchange,change: REAL;
           newrow,uprow,downrow: rowtype;
           done: BOOLEAN;

BEGIN
    newrow[0]:= myrow[0]; newrow[n+1]:= myrow[n+1];
    IF me= 1 THEN downrow:= downchan[me];
    IF me= n THEN uprow:= upchan[me];
```

Jacobi Relaxation (M.II)

```
REPEAT
  FOR k:= 1 TO numiter DO BEGIN(*Αρκετές επαναλήψεις πριν τον έλεγχο τερματισμού*)
    IF me > 1 THEN
      upchan[me-1]:= myrow; (*Αποστολή στον γείτονα me-1*)
      IF me < n THEN BEGIN
        downchan[me+1]:= myrow; (*Αποστολή στον γείτονα me+1*)
        uprow:= upchan[me]; (*Λήψης της νέας uprow*)
      END;
      IF me > 1 THEN
        downrow:= downchan[me]; (*Λήψη της νέας downrow*)
        maxchange:= 0;
        FOR j:= 1 TO n DO BEGIN
          (*Υπολογισμός του μέσου όρου των γειτονικών σημείων*)
          newrow[j]:= (myrow[j-1]+myrow[j+1]+downrow[j]+uprow[j])/4;
          change:= ABS(newrow[j]-myrow[j]);
          IF change > maxchange THEN maxchange:= change;
        END;
        myrow:=newrow;
      END;
    done:= Aggregate(maxchange < tolerance); (*Έλεγχος τερματισμού*)
  UNTIL done;
  out:= myrow; (*Εγγραφή των τελικών αποτελεσμάτων στο F*)
END;

BEGIN
... (*Αρχικοποίηση των τιμών για τον πίνακα A*)
(*Αρχικοποίηση του πίνακα GrayCode για τον Υπερκύβο*)
downchan[1]:= A[0]; upchan[n]:= A[n+1]; (*Καθορισμένες οριακές τιμές
FORALL i:= 1 TO n DO
  (@GrayCode[i] PORT upchan[i]; downchan[i]; inchan[GrayCode[i]])  Updaterow(i, A[i], A[i]);
END.
```

Συνάρτηση Πολλαπλής Συλλογής

Aggregate

```
FUNCTION Aggregate(mydone: BOOLEAN): BOOLEAN;
VAR      mynum, partner, bitvalue, stage: INTEGER;
         hisdone: BOOLEAN;
BEGIN
  mynum:=%SELF; (*Λήψη του αριθμού επεξεργαστή*)
  bitvalue:= 1;
  FOR stage:= 1 TO d DOBEGIN
    IF mynum DIV bitvalue MOD 2 = 0 (*Υπολογισμός του ζεύγους*) THEN partner:= mynum+bitvalue
    ELSE partner:= mynum-bitvalue;
    inchan[partner, stage]:= mydone; (*Αποστολή του mydone στο ταίρι*)
    hisdone:= inchan[mynum, stage]; (*Λήψη του hisdone από το ταίρι*)
    mydone:= mydone AND hisdone;
    bitvalue:= 2*bitvalue (*Μετατόπιση προς την επόμενη θέση bit*)
  END;
  Aggregate:= mydone;
END;
```