ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

ΠΡΟΗΓΜΕΝΑ ΘΕΜΑΤΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΥΠΟΛΟΓΙΣΤΩΝ

Εξετάσεις Σεπτεμβρίου 2010
Διάρκεια 2:45' ώρες

Οι εξετάσεις θα πραγματοποιηθούν ΧΩΡΙΣ την παρουσία βιβλίων, βοηθημάτων ή άλλου είδους σημειώσεων. Το μόνο που επιτρέπεται να έχετε μαζί σας είναι ένα φύλλο Α4 στο οποίο μπορείτε να έχετε γράψει ό,τι έχετε κρίνει πιο σημαντικό για το μάθημα και θέλετε να το έχετε ως βοήθημά σας. Απαγορεύεται η ανταλλαγή οποιουδήποτε αντικειμένου κατά την ώρα της εξέτασης, ούτε και των φύλλων Α4 που είναι ατομικά.

Θέμα 1ο (15%)

Α. *Δώστε τους ορισμούς για τις RAW, WAR, WAW εξαρτήσεις καθώς και ένα παράδειγμα κώδικα για κάθε μια. Με ποιον τρόπο επιλύονται οι αντίστοιχοι hazards στις αρχιτεκτονικές που χρησιμοποιούν τον αλγόριθμο Tomasulo με ROB;*

RAW: Όταν μια εντολή παράγει μια τιμή που χρησιμοποιείται ως όρισμα από μια επόμενη.

```
add   $r1, $r2, $r3
sub   $r5, $r1, $r6
```

WAR: Όταν μια μεταγενέστερη εντολή γράφει σε ένα καταχωρητή που πρέπει να διαβαστεί από μια προγενέστερη εντολή.

```
sub   $r5, $r1, $r6
add   $r1, $r2, $r3
```

WAW: Όταν 2 εντολές γράφουν στον ίδιο καταχωρητή. Αν η εγγραφή γίνει out-of-order τότε ο καταχωρητής θα έχει τη λάθος τιμή.

```
add   $r1, $r2, $r3
sub   $r1, $r4, $r5
```

Ο Tomasulo επιλύει τους WAR και WAW hazards με register renaming. Οι RAW επιλύονται με stalls, μέσω των RSs όπου οι εντολές περιμένουν να παραχθούν τα ορίσματα που χρειάζονται και forwarding των τιμών μέσω του CDB. (2%)

Β. *Αναφέρατε δύο λόγους για τους οποίους η τεχνική βελτιστοποίησης loop unrolling βελτιώνει την απόδοση και ένα λόγο για τον οποίο μπορεί να τη μειώσει.*

Βελτίωση απόδοσης :

- Λιγότερα loop conditional evaluations
- Λιγότερα branches/jumps
- Μεγαλύτερη δυνατότητα για reordering εντολών που ανήκουν σε διαφορετικά iterations
- Δυνατότητα για συγχώνευση loads και stores που ανήκουν σε διαφορετικά iterations

Μείωση απόδοσης :

- Αυξημένος φόρτος (pressure) στην I-cache

Γ. *Δίνεται ο παρακάτω κώδικας γραμμένος σε C :*

```
int touch_array(long *array, long size, long stride, long iter)
```

```
{ long i, sum=0;
  long *p, *limit;
  limit = &(array[size-1]);
  for (i = 0; i < iter; i++)
      for (p = array; p < limit; p+= stride)
            sum += *p;
  returnsum;
}
```

Ο κώδικας έχει μεταγλωττιστεί με τέτοιο τρόπο ώστε οι βαθμωτές μεταβλητές να αποθηκεύονται σε καταχωρητές ενώ το overhead των εντολών ελέγχου του loop (reads και writes) έχει απομακρυνθεί. Ο κώδικας εκτελέστηκε για διαφορετικές τιμές του size και του stride. Η παρακάτω γραφική δείχνει το χρόνο εκτέλεσης του προγράμματος για κάθε περίπτωση.

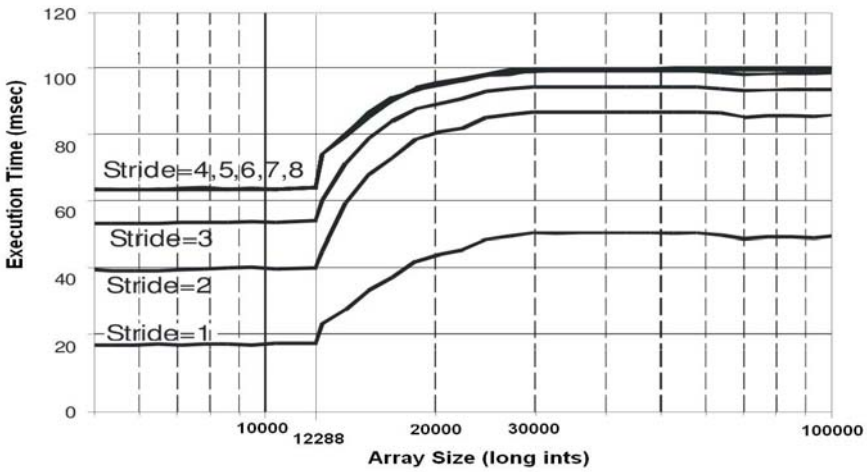
***(i)** Ποιο είναι το μέγεθος της cache σε KB; Δικαιολογήστε την απάντηση σας.*

Όταν ο πίνακας έχει 12288 στοιχεία η απόδοση πέφτει ενώ μέχρι εκείνο το σημείο διατηρείται σταθερή. Επομένως το μέγεθος της cache είναι 12288 * 8 bytes = 96KB.

***(ii)** Ποιο είναι το μέγεθος του cache bock της cache σε bytes; Δικαιολογήστε την απάντηση σας.*

Για stride 1,2,3 η απόδοση είναι καλύτερη από ότι για stride 4, ενώ από εκεί και πέρα δεν υπάρχει μεταβολή. Αυτό σημαίνει ότι για 1,2,3 υπάρχουν πολλαπλά hits σε κάθε block, τα οποία και εξαφανίζονται για stride ≥ 4.

Επομένως το block size είναι 4 * 8 = 32 bytes.



Θέμα 2ο (20%)

***A.** Δίνεται ένας Tournament Branch Predictor, ο οποίος αποτελείται από τα εξής 3 δομικά στοιχεία.*

- *P1: Ένας (1,1) global history predictor, με όλους τους προβλέπτες αρχικοποιημένους στο NT.*
- *P2: Ένας πίνακας με 2-bit predictors, με όλους τους προβλέπτες αρχικοποιημένους στο 01.*
- *P3: Ένας πίνακας με 1-bit predictors, ο καθένας από τους οποίους δείχνει ποιος από τους P1, P2 θα χρησιμοποιηθεί για την πρόβλεψη του συγκεκριμένου άλματος. Όλοι οι προβλέπτες είναι αρχικοποιημένοι να δείχνουν τον P2.*

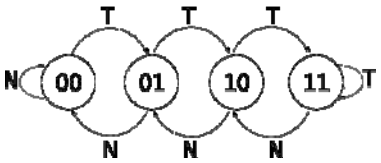
Ο Tournament Predictor προβλέπει το αποτέλεσμα της εντολής άλματος με βάση την πρόβλεψη του predictor που επιλέγει ο P3. Αν η πρόβλεψη αποδειχτεί ότι ήταν λάθος ενώ ο μη επελεγμένος predictor είχε προβλέψει σωστά, τότε ο P3 ενημερώνεται ώστε την επόμενη φορά να επιλέξει τον άλλο predictor. Αυτή είναι η μόνη περίπτωση στην οποία αλλάζει η κατάσταση του P3. Ο κώδικας που εκτελείται περιέχει 3 εντολές άλματος τις Branch1, Branch2 και Branch3. Ο παρακάτω πίνακας δείχνει τη σειρά με την οποία

εκτελούνται οι εντολές άλματος καθώς και το αποτέλεσμα κάθε μιας. Χρησιμοποιήστε έναν πίνακα όπως αυτός, για να δείξετε την πρόβλεψη που κάνει κάθε φορά ο Tournament Predictor καθώς και τη νέα τιμή των P1, P2, P3 με βάση τα εξής :

- *Η τιμή του P1 κωδικοποιείται ως X/Y όπου:*
  - *X – η κατάσταση του 1-bit προβλέπτη που χρησιμοποιείται στην περίπτωση που η τελευταία εντολή άλματος ήταν Not Taken.*
  - *Y - η κατάσταση του 1-bit προβλέπτη που χρησιμοποιείται στην περίπτωση που η τελευταία εντολή άλματος ήταν Taken..*
- *Η τιμή του P2 κωδικοποιείται ως 00, 01, 10, ή 11.*
- *Η τιμή του P3 είναι είτε P1 είτε P2.*

Branch	Αποτέλεσμα	Πρόβλεψη	Νέα τιμή του P1		Νέα τιμή του P2		Νέα τιμή του P3	
			Branch2	Branch3	Branch2	Branch3	Branch2	Branch3
Branch1	NT							
Branch2	T	N	T/N	N/N	10	01	P2	P2
Branch3	T	N	T/N	N/T	10	10	P2	P2
Branch1	T							
Branch2	NT	T	T/N	N/T	01	10	P1	P2
Branch3	T	T	T/N	T/T	01	11	P1	P2
Branch1	NT							
Branch2	T	T	T/N	T/T	10	11	P1	P2
Branch3	T	T	T/N	T/T	10	11	P1	P2
Branch1	T							
Branch2	NT	N	T/N	T/T	01	11	P1	P2
Branch3	T	T	T/N	T/T	01	11	P1	P2

Δε χρειάζεται να συμπληρώσετε τις γραμμές που αντιστοιχούν στο Branch1. Αυτές οι γραμμές δίνονται ώστε να φαίνεται η σειρά εκτέλεσης των εντολών και να έχετε το σωστό branch history για τον P1 προβλέπτη. Ο P2 χρησιμοποιεί το εξής FSM :



***B.** Σαν αρχιτέκτονες επιλέγετε να χρησιμοποιείτε ένα πίνακα από n-bit προβλέπτες αντί για ένα μοναδικό προβλέπτη τον οποίο μοιράζονται όλες οι εντολές άλματος. Γιατί; Δώστε μια ακολουθία αλμάτων όπου ο ρυθμός αποτυχημένων προβλέψεων είναι μικρότερος όταν 2 άλματα μοιράζονται τον ίδιο 1-bit προβλέπτη σε σχέση με την περίπτωση όπου κάθε ένα από τα 2 άλματα χρησιμοποιεί τον δικό του ξεχωριστό 1-bit προβλέπτη.*

Έστω 2 branches B1, B2 που εκτελούνται συνεχώς το ένα μετά το άλλο και μοιράζονται ένα 1 bit predictor, αρχικοποιημένο στο NT.

Εντολή	B1	B2	B1	B2	B1	B2	B1	B2
Αποτέλεσμα	T	NT	NT	T	T	NT	NT	T
Πρόβλεψη	NT	T	NT	NT	T	T	NT	NT
Σωστή	όχι	όχι	ναι	όχι	ναι	όχι	ναι	όχι

Ποσοστό επιτυχίας 3/8.

Έστω τώρα ότι το κάθε branch έχει το δικό του 1 bit predictor. Για την ίδια ακολουθία θα είχαμε :

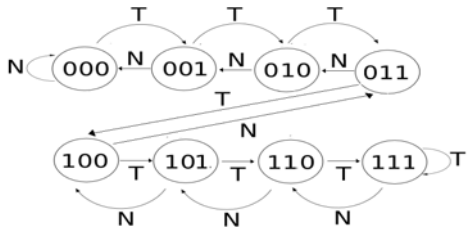
Εντολή	B1	B2	B1	B2	B1	B2	B1	B2
Αποτέλεσμα	T	NT	NT	T	T	NT	NT	T
Πρόβλεψη	NT	NT	T	NT	NT	T	T	NT
Σωστή	όχι	ναι	όχι	όχι	όχι	όχι	Όχι	όχι

Ποσοστό επιτυχίας 1/8.

Θέμα 3ο (30%)

Δίνεται αρχιτεκτονική η οποία υλοποιεί τον αλγόριθμο Tomasulo χρησιμοποιώντας ROB για in-order commit εντολών. Το pipeline του επεξεργαστή περιέχει τα στάδια Issue (IS), Execute (EX), Write Result (WR) και Commit (CMT), αγνοούμε δηλαδή τα IF και ID. Ισχύουν επίσης τα ακόλουθα :

- *Τα IS, WR, CMT απαιτούν 1 κύκλο.*
- *Το σύστημα περιέχει άπειρο αριθμό από reservation stations (RS).*
- *Το σύστημα περιλαμβάνει 2 non-pipelined integer functional unit. Στο ένα εκτελούνται προσθέσεις/αφαιρέσεις ακεραίων, οι οποίες διαρκούν 1 κύκλο, καθώς και η αφαίρεση που απαιτούν οι εντολές άλματος υπό συνθήκη. Στο δεύτερο εκτελούνται πολλαπλασιασμοί/διαιρέσεις ακεραίων, οι οποίοι διαρκούν 5 κύκλους. Η εντολή διαίρεσης ακεραίων αποθηκεύει το πηλίκο στον καταχωρητή Lo και το υπόλοιπο στον καταχωρητή Hi, από όπου μπορεί να μεταφερθεί με την εντολή MFHI. Η εντολή αυτή αντιμετωπίζεται από το σύστημα σαν μια απλή αριθμητική εντολή ακεραίων, διαρκεί ένα κύκλο και δεν απαιτεί την χρήση του integer functional unit.*
- *Το σύστημα περιλαμβάνει 2 non-pipelined floating point functional units, ένα για ADDD / SUBD και ένα για MULD / DIVD. Οι εντολές πρόσθεσης/αφαίρεσης διαρκούν 2 κύκλους, ενώ οι εντολές πολλαπλασιασμού/διαίρεσης 7 κύκλους.*
- *Για τις εντολές αναφοράς στη μνήμη, στο στάδιο EX γίνεται τόσο ο υπολογισμός της διεύθυνσης αναφοράς όσο και η προσπέλαση στη μνήμη. Το σύστημα περιλαμβάνει ένα Load και ένα Store Queue, τα οποία διαθέτουν 4 θέσεις. Οι εντολές χρησιμοποιούν ένα ξεχωριστό functional unit για τον υπολογισμό της διεύθυνσης και διαρκούν 2 κύκλους σε περίπτωση Hit στην cache και 6 κύκλους σε περίπτωση Miss.*
- *Ο ROB έχει 8 θέσεις.*
- *Το σύστημα περιλαμβάνει 1 CDB. Σε περίπτωση που παραπάνω από μια εντολές θέλουν να το χρησιμοποιήσουν, τότε προτεραιότητα αποκτά η ''παλαιότερη'' εντολή.*
- *Για τις εντολές διακλάδωσης υπό συνθήκη, το σύστημα χρησιμοποιεί έναν 3-bit predictor με συνολικά 16 εγγραφές (entries). Οι 3-bit predictors είναι αρχικοποιημένοι στο 000. Δίνεται το παρακάτω FSM του 3-bit predictor :*



- *Η πρόβλεψη για μια εντολή διακλάδωσης υπό συνθήκη γίνεται ταυτόχρονα με τη δρομολόγηση της εντολής. Ο έλεγχος της πρόβλεψης γίνεται αμέσως μόλις γίνει γνωστό το αποτέλεσμα της εντολής, δηλαδή στο στάδιο WR (κύκλος k). Σε περίπτωση σφάλματος, σταματά η εκτέλεση των λάθος εντολών και στον επόμενο κύκλο (κύκλος k+1) δρομολογείται η σωστή εντολή.*

Δίνεται ο παρακάτω κώδικας :

```
0x00880000        LOOP:       LD    F2, 0(R2)
0x00880004                    ADDD  F4, F2, F3
0x00880008                    MULD  F6, F2, F4
0x0088000C                    ADDD  F0, F0, F6
0x00880010                    DIVI  R1, #2            // R4 = R1 % 2;
0x00880014                    MFHI  R4
0x00880018                    BNEZ  R4, L1
0x0088001C                    ADDI  R2, R2, #8
0x00880020                    LD    F2, 0(R2)
0x00880024                    MULD  F2, F2, F6
0x00880028                    ADDD  F0, F0, F2
0x0088002C        L1:         ADDI  R2, R2, #8
0x00880030                    ADDI  R1, R1, #1
0x00880034                    SUBI  R5, R5, #1
0x00880038                    BNEZ  R5, LOOP
0x0088003C                    ADDI  R2, R2, #8
0x00880040                    SD    F0, 0(R2)
0x00880044                    ADDI  R2, R2, #8
```

Ο καταχωρητής R2 περιέχει τη διεύθυνση του πρώτου στοιχείου ενός πίνακα Α, στον οποίο είναι αποθηκευμένοι αριθμοί διπλής ακρίβειας (μεγέθους 8 bytes ο καθένας). Η cache είναι fully associative απείρου μεγέθους, ενώ το μέγεθος του cache block είναι 8 bytes. Στην αρχή η cache είναι άδεια. Δίνονται επίσης οι αρχικές τιμές 1 και 2 για τους καταχωρητές R1 και R5 αντίστοιχα. Εκτελέστε τον παραπάνω κώδικα και δώστε τους χρόνους δρομολόγησης, εκτέλεσης και ολοκλήρωσης των εντολών σε έναν πίνακα όπως ο παρακάτω :

Αφού ο πίνακας έχει 16 εγγραφές, θα χρησιμοποιούνται τα 4 τελευταία bits του PC των branches για να προσπελαύνεται ο αντίστοιχος predictor. Επομένως και τα 2 branches του κώδικα χρησιμοποιούν τον ίδιο predictor.

OP		IS	EX	WR	CMT	Σχόλιο
L.D	F2, 0(R2)	1	2 - 7	8	9	Cache Miss
ADDD	F4, F2, F3	2	9 - 10	11	12	RAW (F2)
MULD	F6, F2, F4	3	12 - 18	19	20	RAW (F4)
ADDD	F0, F0, F6	4	20 - 21	22	23	RAW (F6)
DIVI	R1,#2	5	6 – 10	12	24	CDB conflict
MFHI	R4	6	13	14	25	RAW (Hi)
BNEZ	R4, L1	7	15	16	26	pred = NT, res = T new_val = 001, flush @16
ADDI	R2, R2, #8	8	9 - 9	10		
L.D	F2, 0(R2)	10	11 - 16			ROB full, Cache miss
MULD	F2, F2, F6	13				ROB full, RAW (F2)
ADDI	R2, R2, #8	17	18	20	27	CDB conflict
ADDI	R1, R1, #1	18	19	21	28	CDB conflict
SUBI	R5, R5, #1	19	20	23	29	CDB conflict
BNEZ	R5, LOOP	21	24	25	30	RAW (R5), pred = NT, res = T new_val = 010, flush @25
ADDI	R2, R2, #8	24	25			
SD	F0, 0(R2)	25				
L.D	F2, 0(R2)	26	27 - 28	29	31	Cache Hit
ADDD	F4, F2, F3	27	30 - 31	32	33	RAW (F2)
MULD	F6, F2, F4	28	33 - 39	40	41	RAW (F4)
ADDD	F0, F0, F6	29	41 - 42	43	44	RAW (F6)
DIVI	R1,#2	30	31 - 35	36	45	
MFHI	R4	31	37	38	46	RAW (Hi)
BNEZ	R4, L1	32	39	41	47	CDB conflict, pred = NT, res = NT, new_val = 001
ADDI	R2, R2, #8	33	34	35	48	
L.D	F2, 0(R2)	34	36 - 41	42	49	Cache Miss
MULD	F2, F2, F6	35	43 - 49	50	51	RAW (F2)
ADDD	F0, F0, F2	42	51 - 52	53	54	ROB full, RAW(F2)
ADDI	R2, R2, #8	45	46 – 46	47	55	
ADDI	R1, R1, #1	46	47	48	56	
SUBI	R5, R5, #1	47	48	49	57	
BNEZ	R5, LOOP	48	50	51	58	pred = NT, res = NT, new_val = 000
ADDI	R2, R2, #8	49	51	52	59	struct. hazard
SD	F0, 0(R2)	50	53 - 58	59	60	RAW (R2), Cache Miss
ADDI	R2, R2, #8	52	53	54	61	ROB full

Θέμα 4ο (25%)

Θεωρούμε το ακόλουθο κομμάτι κώδικα:

```
float x[256][256], y[256][256];
int i,j;
for(i=0; i<16; i++)
      for(j=0; j<256; j++)
            x[i][j] += y[i][j] + y[i+1][j];
```

Κάνουμε τις εξής υποθέσεις:

1. *Το πρόγραμμα εκτελείται σε έναν επεξεργαστή με μόνο ένα επίπεδο κρυφής μνήμης δεδομένων, η οποία αρχικά είναι άδεια. Η κρυφή μνήμη είναι direct-mapped, write-allocate, και έχει μέγεθος 2KB. Το μέγεθος του block είναι 32 bytes, ενώ το μέγεθος ενός float είναι 4 bytes.*
2. *Η σειρά με την οποία γίνονται οι αναγνώσεις είναι x[i][j], y[i][j], y[i+1][j].*
3. *Δήλωση διαδοχικών μεταβλητών συνεπάγεται αποθήκευση τους σε συνεχόμενες θέσεις μνήμης.*
4. *Οι πίνακες είναι ευθυγραμμισμένοι και αποθηκευμένοι κατά γραμμές.*

Α. *Βρείτε το συνολικό ποσοστό αστοχίας (miss rate) για τις αναφορές που γίνονται στην μνήμη.*

Η cache έχει $(2 * 2^{10}) / 2^5 = 2^6 = 64$ blocks. Το κάθε block χωράει 8 στοιχεία από τον κάθε πίνακα. Η κάθε γραμμή των πινάκων καταλαμβάνει 256 * 4 = 1KB ή αλλιώς 32 blocks της cache. Επομένως η γραμμή x_0 θα καταλαμβάνει τα 32 blocks της cache και η x_1 τα επόμενα 32. Η x_2 θα καταλαμβάνει τα ίδια blocks με την x_0, η x_3 με τη x_1, …., η x_{255} με τη x_1. Έτσι η y_0 θα καταλαμβάνει τα ίδια blocks με τη x_0 κτλ. αφού οι 2 πίνακες είναι αποθηκευμένοι σε συνεχόμενες θέσεις της μνήμης.

x[0][0]	y[0][0]	y[1][0]	x[0][0]	m m m m	comp	comp	comp	conf
x[0][1]	y[0][1]	y[1][1]	x[0][1]	h m h m		conf		conf
x[0][2]	y[0][2]	y[1][2]	x[0][2]	h m h m		conf		conf
…….								
x[0][7]	y[0][7]	y[1][7]	x[0][7]	h m h m		conf		conf
x[0][8]	y[0][8]	y[1][8]	x[0][8]	m m m m	comp	comp	comp	conf
x[0][9]	y[0][9]	y[1][9]	x[0][9]	h m h m		conf		conf
x[0][10]	y[0][10]	y[1][10]	x[0][10]	h m h m		conf		conf
……...								
x[0][15]	y[0][15]	y[1][15]	x[0][15]	h m h m		conf		conf
……..								
x[0][248]	y[0][248]	y[1][248]	x[0][248]	m m m m	comp	comp	comp	conf
x[0][249]	y[0][249]	y[1][249]	x[0][249]	h m h m		conf		conf
x[0][250]	y[0][250]	y[1][250]	x[0][250]	h m h m		conf		conf
…….								
x[0][255]	y[0][255]	y[1][255]	x[0][255]	h m h m		conf		conf

Στο τέλος του iteration για i=0, η cache έχει τα εξής περιεχόμενα :

X[0][0] – X[0][7]
X[0][8] – X[0][15]
X[0][16] – X[0][23]
…….
X[0][248] – X[0][255]

Y[1][0] – Y[1][7]
Y[1][8] – Y[1][15]
Y[1][16] – Y[1][23]
……
Y[1][248] – Y[1][255]

Με βάση τα mappings, η γραμμή x_1 θα αντιστοιχίζεται στα ίδια blocks με τη γραμμή y_1. Έτσι, θα έχουμε :

x[1][0]	y[1][0]	y[2][0]	x[1][0]	m m m m	comp	conf	comp	conf
x[1][1]	y[1][1]	y[2][1]	x[1][1]	h m h m		conf		conf
x[1][2]	y[1][2]	y[2][2]	x[1][2]	h m h m		conf		conf
…….								
x[1][7]	y[1][7]	y[2][7]	x[1][7]	h m h m		conf		conf
x[1][8]	y[1][8]	y[2][8]	x[1][8]	m m m m	comp	conf	comp	conf
x[1][9]	y[1][9]	y[2][9]	x[1][9]	h m h m		conf		conf
x[1][10]	y[1][10]	y[2][10]	x[1][10]	h m h m		conf		conf
…….								
x[1][15]	y[1][15]	y[2][15]	x[1][15]	h m h m		conf		conf
……..								
x[1][248]	y[1][248]	y[2][248]	x[1][248]	m m m m	comp	conf	comp	conf
x[1][249]	y[1][249]	y[2][249]	x[1][249]	h m h m		conf		conf
x[1][250]	y[1][250]	y[2][250]	x[1][250]	h m h m		conf		conf
…….								
x[1][255]	y[1][255]	y[2][255]	x[1][255]	h m h m		conf		conf

Επομένως συνολικά έχουμε misses = 16 * 32 * 18 = 9216 και miss_rate = 9216/16384 = 56.25%

B. *Ποιες από τις παρακάτω τεχνικές βελτιστοποίησης θα ακολουθούσατε προκειμένου να βελτιώσετε την απόδοση; Δικαιολογήστε την επιλογή ή μη-επιλογή κάθε τεχνικής. Για αυτές που επιλέξατε, υπολογίστε το νέο ποσοστό αστοχίας.*

1. *Αύξηση block size στα 64 bytes (με διατήρηση της χωρητικότητας της cache)*

Διπλασιάζοντας το μέγεθος του block διατηρώντας ταυτόχρονα σταθερή τη χωρητικότητα της cache, μειώνουμε στο μισό το πλήθος των blocks. Έτσι, πλέον η cache διαθέτει 32 blocks, κάθε ένα εκ των οποίων χωρά 16 στοιχεία από τον κάθε πίνακα. Αντίστοιχα επηρεάζονται και τα mappings, αφού τώρα κάθε γραμμή του πίνακα απαιτεί 16 αντί για 32 blocks. Τα conflicts όμως παραμένουν τα ίδια.

Όσον αφορά το pattern των προσβάσεων στη μνήμη, για κάθε 16-άδα θα γλυτώνουμε 2 compulsory misses του x_i και του y_{i+1}.Έτσι τα misses θα είναι τώρα

misses = 16 * 16 * (18+16) = 8704 και miss_rate = 53.125%

Αφού ο ρυθμός αστοχίας μειώνεται θα την επιλέγαμε.

2. *Αύξηση associativity σε 2-way (με διατήρηση της χωρητικότητας της cache)*

Διπλασιάζοντας το associativity διατηρώντας ταυτόχρονα σταθερή τη χωρητικότητα της cache καταλήγουμε σε ένα σχήμα με 32 cache lines, καθεμιά από τις οποίες περιέχει 2 cache blocks. Αφού η

κάθε γραμμή του πίνακα καταλαμβάνει 32 cache blocks, συμπεραίνουμε πως ανά μια γραμμή θα έχουμε conflicts. Έτσι (υποθέτοντας LRU πολιτική) το pattern τώρα θα είναι :

```
x[0][0]   y[0][0]   y[1][0]   x[0][0]        m m m m    comp  comp  comp  conf
x[0][1]   y[0][1]   y[1][1]   x[0][1]        h  m m  m          conf  conf  conf
x[0][2]   y[0][2]   y[1][2]   x[0][2]        h  m m  m          conf  conf  conf
……..
x[0][7]   y[0][7]   y[1][7]   x[0][7]        h  m m  m          conf  conf  conf

x[0][8]   y[0][8]   y[1][8]    x[0][8]       m m m m    comp  comp  comp  conf
x[0][9]   y[0][9]   y[1][9]    x[0][9]       h  m m  m          conf  conf  conf
x[0][10] y[0][10]  y[1][10]  x[0][10]       h  m m  m          conf  conf  conf
……...
x[0][15] y[0][15]  y[1][15]  x[0][15]       h  m m  m          conf  conf  conf


……..

x[0][248]   y[0][248]   y[1][248]   x[0][248]    m m m m    comp  comp  comp  conf
x[0][249]   y[0][249]   y[1][249]   x[0][249]    h  m m  m          conf  conf  conf
x[0][250]   y[0][250]   y[1][250]    x[0][250]   h  m m  m          conf  conf  conf
……..
x[0][255]  y[0][255]  y[1][255]     x[0][255]   h  m m  m          conf  conf  conf
```

Στο τέλος του iteration για i=0, η cache έχει τα εξής περιεχόμενα :

X[0][0] – X[0][7]	Y[1][0] – Y[1][7]
X[0][8] – X[0][15]	Y[1][8] – Y[1][15]
X[0][16] – X[0][23]	Y[1][16] – Y[1][23]
……	……
X[0][248] – X[0][255]	Y[1][248] – Y[1][255]

όπου το LRU block κάθε cache line είναι αυτό που περιέχει τον πίνακα y. Έτσι για i = 1 θα έχουμε :

```
x[1][0]   y[1][0]   y[2][0]   x[1][0]        m m m m    comp  conf  comp  conf
x[1][1]   y[1][1]   y[2][1]   x[1][1]        h  m m  m          conf  conf  conf
x[1][2]   y[1][2]   y[2][2]   x[1][2]        h  m m  m          conf  conf  conf
…….
x[1][7]   y[1][7]   y[2][7]   x[1][7]        h  m m  m          conf  conf  conf

x[1][8]   y[1][8]   y[2][8]    x[1][8]       m m m m    comp  conf  comp  conf
x[1][9]   y[1][9]   y[2][9]    x[1][9]       h  m m  m          conf  conf  conf
x[1][10] y[1][10]  y[2][10]  x[1][10]       h  m m  m          conf  conf  conf
…….
x[1][15] y[1][15]  y[2][15]  x[1][15]       h  m m  m          conf  conf  conf


……..

x[1][248]   y[1][248]   y[2][248]   x[1][248]    m m m m    comp  conf  comp  conf
x[1][249]   y[1][249]   y[2][249]   x[1][249]    h  m m  m          conf  conf  conf
x[1][250]   y[1][250]   y[2][250]    x[1][250]   h  m m  m          conf  conf  conf
……..
x[1][255]  y[1][255]  y[2][255]     x[1][255]   h  m m  m          conf  conf  conf
```

Επομένως misses = 16 * 32 * 25 = 12800 και miss_rate = 78.125%. Άρα δε θα επιλέγαμε τη τεχνική αυτή.

3. *Loop distribution*

Ο κώδικας μπορεί να γραφτεί ως εξής :

```
for(i = 0; i < 16; i++)
    for(j = 0; j < 256; j++)
        x[i][j] += y[i][j]

for(i = 0; i < 16; i++)
    for(j = 0; j < 256; j++)
        x[i][j] += y[i+1][j]
```

Για το πρώτο loop έχουμε :

x[0][0]	y[0][0]	x[0][0]	m m m	comp	comp conf
x[0][1]	y[0][1]	x[0][1]	h m m		conf conf
x[0][2]	y[0][2]	x[0][2]	h m m		conf conf
…….					
x[0][7]	y[0][7]	x[0][7]	h m m		conf conf
x[0][8]	y[0][8]	x[0][8]	m m m	comp	comp conf
x[0][9]	y[0][9]	x[0][9]	h m m		conf conf
x[0][10]	y[0][10]	x[0][10]	h m m		conf conf
……...					
x[0][15]	y[0][15]	x[0][15]	h m m		conf conf

misses = 16 * 32 * (3+14) = 8704, accesses = 16 * 32 *24 = 12288

Για το δεύτερο loop έχουμε :

x[0][0]	y[1][0]	x[0][0]	m m h	comp	comp
x[0][1]	y[1][1]	x[0][1]	h h h		
x[0][2]	y[1][2]	x[0][2]	h h h		
…….					
x[0][7]	y[1][7]	x[0][7]	h h h		
x[0][8]	y[1][8]	x[0][8]	m m h	comp	comp
x[0][9]	y[1][9]	x[0][9]	h h h		
x[0][10]	y[1][10]	x[0][10]	h h h		
……...					
x[0][15]	y[1][15]	x[0][15]	h h h		

misses = 16 * 32 * 2 = 1024, accesses = 16 * 32 *24 = 12288

Άρα miss_rate = 39.5%. Αυτός ο ρυθμός προκύπτει από την αύξηση των misses από 9216 σε 9728 και των συνολικών προσβάσεων στην ιεραρχία μνήμης από 16384 σε 24756 (αύξηση 50%). Επομένως, δε θα επιλέγαμε αυτή την τεχνική.