Πανεπιστήμιο Θεσσαλίας
Τμήμα Πληροφορικής

Αρχιτεκτονική Υπολογιστών

Λυμένες Ασκήσεις

Άσκηση 4:

Έστω ένας επεξεργαστής βασισμένος στην αρχιτεκτονική του πίνακα παρακολούθησης εντολών (scoreboard) για εκτέλεση εκτός σειράς. Στη ΜΕΔ του επεξεργαστή αυτού περιλαμβάνονται δύο υπομονάδες πρόσθεσης/αφαίρεσης κινητής υποδιαστολής (fpadd), δύο υπομονάδες πολλαπλασιασμού/διαίρεσης κινητής υποδιαστολής (fpmul) και τρεις υπομονάδες ακέραιων λογικών/αριθμητικών πράξεων (int). Οι χρόνοι εκτέλεσης των αντίστοιχων εντολών είναι οι εξής:

Εντολή	*Χρόνος εκτέλεσης*
πρόσθεση/αφαίρεση κινητής υποδιαστολής	2cc
πολλαπλασιασμός κινητής υποδιαστολής	8cc
διαίρεση κινητής υποδιαστολής	20cc
ακέραια πράξη	1cc
προσπέλαση μνήμης	2cc

όπου οι εντολές προσπέλασης μνήμης εκτελούνται στις υπομονάδες ακέραιων πράξεων, ενώ καμία υπομονάδα δεν υποστηρίζει επικαλυπτόμενες λειτουργίες.
Αν ο επεξεργαστής δέχεται το σύνολο εντολών MIPS, δείξτε σε έναν πίνακα τους χρόνους κατά τους οποίους κάθε εντολή του παρακάτω βρόχου εκδίδεται σε κάποια υπομονάδα εκτέλεσης, διαβάζει τα τελούμενα εισόδου, αρχίζει την εκτέλεσή της και γράφει το αποτέλεσμά της στο φάκελο καταχωρητών, για τις τρεις πρώτες επαναλήψεις του βρόχου, αν ο επεξεργαστής είναι (α) βαθμωτός, και (β) υπερβαθμωτός βαθμού 2.

```
loop: ldc1  $f6,32($2)        (S1)
      ldc1  $f2,40($3)        (S2)
      div.d $f0,$f2,$f4       (S3)
      sub.d $f8,$f6,$f2       (S4)
      mul.d $f10,$f8,$f6      (S5)
      add.d $f6,$f0,$f10      (S6)
      sdc1  $f6,32($2)        (S7)
      addiu $2,$2,8           (S8)
      addiu $3,$3,8           (S9)
      bne   $2,$10,loop       (S10)
```

Σε κάθε περίπτωση μη ιδανικού χρονισμού των εντολών, να εξηγήσετε το λόγο της καθυστέρησης. Υποθέστε ότι ο φάκελος καταχωρητών είναι ενιαίος με 32 καταχωρητές σταθερής και 32 καταχωρητές κινητής υποδιαστολής, και με όσες θύρες ανάγνωσης/εγγραφής απαιτεί η αρχιτεκτονική – για εξυπηρέτηση μιας εντολής στη βαθμωτή και δύο εντολών στην υπερβαθμωτή ανά κύκλο μηχανής, και ότι η μνήμη υποστηρίζει μία προσπέλαση δεδομένων ανά κύκλο μηχανής, χωρίς δυνατότητα προσπέλασης εκτός σειράς. Υποθέστε επιτυχημένη πρόβλεψη διακλαδώσεων, χωρίς όμως υποθετική εκτέλεση εντολών, κι επομένως μια εντολή δε μπορεί να εκτελεστεί, αν προηγείται εντολή διακλάδωσης που εκκρεμεί. Δείξτε τη μορφή που έχουν οι πίνακες δρομολόγησης των εντολών στον κύκλο μηχανής που η εντολή S5 γράφει το αποτέλεσμά της στο φάκελο καταχωρητών για τη δεύτερη επανάληψη του βρόχου. Εκτιμήστε το μέσο ρυθμό ολοκλήρωσης των επαναλήψεων του βρόχου, όπως και το μέσο ρυθμό ολοκλήρωσης εντολών, όταν εκτελείται μεγάλος αριθμός επαναλήψεων, και για τις δύο περιπτώσεις αρχιτεκτονικής.

Απάντηση

Σε έναν επεξεργαστή που εκτελεί εντολές εκτός σειράς με τη βοήθεια πίνακα παρακολούθησης (scoreboard), κάθε εντολή, αφού αποκωδικοποιηθεί και εισαχθεί στον πίνακα, περνάει διαδοχικά από τις ακόλουθες φάσεις:

1. Έκδοση σε κάποια υπομονάδα εκτέλεσης. Απαραίτητες προϋποθέσεις για έκδοση είναι (α) να υπάρχει κατάλληλη ελεύθερη υπομονάδα, και (β) να μην υπάρχει στον πίνακα παρακολούθησης άλλη εντολή που να γράφει το αποτέλεσμά της στον ίδιο καταχωρητή.
2. Ανάγνωση τελούμενων εισόδου από το φάκελο καταχωρητών. Επειδή δεν υπάρχει παροχέτευση, η ανάγνωση των τελούμενων εισόδου γίνεται πάντοτε από το φάκελο καταχωρητών, μετά την εγγραφή τους από προηγούμενες εντολές.
3. Εκτέλεση. Η εκτέλεση μιας εντολής μπορεί να ξεκινήσει, μόλις διαβαστούν τα τελούμενα εισόδου, εφ' όσον δεν προηγείται διακλάδωση που να εκκρεμεί.
4. Εγγραφή αποτελέσματος στο φάκελο καταχωρητών. Η εγγραφή στο φάκελο καταχωρητών μπορεί να γίνει, μόνο αν δεν υπάρχει προηγούμενη εντολή που να διαβάζει τον ίδιο καταχωρητή, και η οποία δεν έχει ολοκληρώσει τη φάση 2.

Για απλούστευση, αγνοούμε το γεγονός της πεπερασμένης χωρητικότητας του πίνακα παρακολούθησης, και υποθέτουμε ότι οι εντολές εισάγονται στον πίνακα χωρίς καθυστέρηση. Επίσης, υποθέτουμε ότι οι παραπάνω φάσεις του κύκλου εντολής – εκτός της φάσης εκτέλεσης – καταλαμβάνουν έναν κύκλο μηχανής, ενώ η προσπέλαση του φακέλου καταχωρητών επιτρέπει εγγραφή στο πρώτο και ανάγνωση στο δεύτερο μισό του κύκλου μηχανής. Τέλος, υποθέτουμε ότι μια υπομονάδα γίνεται διαθέσιμη για εκτέλεση άλλης εντολής στον ίδιο κύκλο μηχανής που γράφει το προηγούμενο αποτέλεσμά της στο φάκελο καταχωρητών, ή, σε περίπτωση που δε γράφει κάποιο αποτέλεσμα στο φάκελο καταχωρητών, στον κύκλο μηχανής που ολοκληρώνει την εκτέλεσή της.
Φυσικά, και οι τέσσερις παραπάνω φάσεις του κύκλου εντολής μπορούν να τροποποιηθούν προς διάφορες κατευθύνσεις, αλλά για την παρούσα άσκηση θα τις χρησιμοποιήσουμε σύμφωνα με το βασικό αλγόριθμο δρομολόγησης εντολών που περιγράφεται στο βιβλίο των Hennessy και Patterson. Θα δείξουμε δε το ζητούμενο χρονισμό των εντολών σε έναν πίνακα ανά περίπτωση, όπου στις δύο πρώτες στήλες θα έχουμε τον αριθμό επανάληψης (iteration) και τον αριθμό εντολής (instruction) αντίστοιχα, στις επόμενες τέσσερις θα έχουμε τους χρόνους έκδοσης (issue), ανάγνωσης τελούμενων (read operands), έναρξης εκτέλεσης (execute) και αποθήκευσης αποτελέσματος (write result) αντίστοιχα, ενώ στην τελευταία στήλη θα έχουμε σχόλια που να εξηγούν την καθυστέρηση, αν υπάρχει.

Έτσι, στην περίπτωση βαθμωτού επεξεργαστή, υποθέτοντας ότι η πρώτη εντολή εκδίδεται στον κύκλο 1, ο χρονισμός των εντολών θα γίνει σύμφωνα με τον πιο κάτω πίνακα:

iter.	instr.	issue	read operands	execute	write result	σχόλια
1	S1	1	2	3	5	int1
1	S2	2	3	4	6	int2
1	S3	3	6	7	27	fpmul1, αναμονή $f2 (AME)
1	S4	4	7	8	10	fpadd1, αναμονή $f2 και ΦΚ
1	S5	5	10	11	19	fpmul2, αναμονή $f8 (AME)
1	S6	6	27	28	30	fpadd2, αναμονή $f0 (AME)
1	S7	7	30	31		int1, αναμονή $f6 (AME)
1	S8	8	9	10	31	int2, αναμονή $2 (EMA)
1	S9	9	11	12	13	int3, αναμονή ΦΚ
1	S10	13	31	32		αναμονή int3, $2 (AME)
2	S1	31	32	33	35	αναμονή $f6 (EME), int2
2	S2	32	33	34	36	int1

2	S3	33	36	37	57	fpmul1, αναμονή $f2 (AME)
2	S4	34	37	38	40	fpadd1, αναμονή $f2 και ΦΚ
2	S5	35	40	41	49	fpmul2, αναμονή $f8 (AME)
2	S6	36	57	58	60	fpadd2, αναμονή $f0 (AME)
2	S7	37	60	61		int1, αναμονή $f6 (AME)
2	S8	38	39	40	61	int2, αναμονή $2 (EMA)
2	S9	39	41	42	43	int3, αναμονή ΦΚ
2	S10	43	61	62		αναμονή int3, $2 (AME)
3	S1	61	62	63	65	αναμονή $f6 (EME), int2
3	S2	62	63	64	66	int1
3	S3	63	66	67	87	fpmul1, αναμονή $f2 (AME)
3	S4	64	67	68	70	fpadd1, αναμονή $f2 και ΦΚ
3	S5	65	70	71	79	fpmul2, αναμονή $f8 (AME)
3	S6	66	87	88	90	fpadd2, αναμονή $f0 (AME)
3	S7	67	90	91		int1, αναμονή $f6 (AME)
3	S8	68	69	70	91	int2, αναμονή $2 (EMA)
3	S9	69	71	72	73	int3, αναμονή ΦΚ
3	S10	73	91	92		αναμονή int3, $2 (AME)

όπου στην τελευταία στήλη δείχνουμε τη συγκεκριμένη υπομονάδα στην οποία εκδίδεται η εντολή, καθώς και τον όποιο λόγο καθυστέρησης σε μια ή περισσότερες από τις φάσεις του κύκλου εντολής, που μπορεί να είναι κάποια εξάρτηση από δεδομένα, ή δομική εξάρτηση στην έκδοση σε κάποια υπομονάδα ή στην προσπέλαση του φακέλου καταχωρητών και της μνήμης. Παρατηρούμε ότι ο κίνδυνος από την εξάρτηση τύπου EME που εμφανίζεται μεταξύ των εντολών S6 και S1 στον καταχωρητή $f6, καθώς και ο κίνδυνος από την εξάρτηση τύπου EMA που εμφανίζεται μεταξύ των εντολών S7 και S8 στον καταχωρητή $2, υποχρεώνουν το βρόχο να εκτελείται σειριακά. Η πρώτη από αυτές μεταφέρεται από επανάληψη σε επανάληψη του βρόχου και εμποδίζει την έκδοση της εξαρτημένης εντολής S1. Η ίδια εξάρτηση, με αντίστροφη όμως κατεύθυνση από την εντολή S1 προς την S6, εμφανίζεται και μεταξύ εντολών της ίδιας επανάληψης, αλλά στην περίπτωση αυτή δεν εμφανίζεται κίνδυνος, λόγω του μεγάλου αριθμού εντολών που μεσολαβεί. Η δεύτερη από τις πιο πάνω εξαρτήσεις μας αναγκάζει να καθυστερήσουμε την αποθήκευση στο φάκελο καταχωρητών της νέας τιμής του καταχωρητή $2, γεγονός που έχει σα συνέπεια την καθυστέρηση στην ανάγνωση τελούμενων της εντολής S10.
Όσο αφορά τις δομικές εξαρτήσεις, παρατηρούμε ότι οι πιο σημαντικές είναι αυτές που σχετίζονται με υπομονάδες εκτέλεσης, μια που αυτές εμποδίζουν την έκδοση εντολών και αναγκάζουν τη ΜΕΔ σε πάγωμα διαρκείας, έως ότου η αντίστοιχη υπομονάδα γίνει διαθέσιμη. Για παράδειγμα, η εντολή S10 δε μπορεί να εκδοθεί αμέσως μετά την S9, επειδή τότε δεν υπάρχει διαθέσιμη υπομονάδα ακέραιων πράξεων. Είναι αξιοσημείωτο ότι η μη διαθεσιμότητα των υπομονάδων δεν οφείλεται πάντα σε μεγάλους χρόνους εκτέλεσης, αλλά και σε άλλους παράγοντες, όπως (α) σε εξαρτήσεις τύπου EMA που δεν επιτρέπουν την απόδοση των υπομονάδων σε άλλες εντολές, παρά το γεγονός ότι οι αντίστοιχες εκτελέσεις έχουν ολοκληρωθεί, κάτι που εδώ συμβαίνει με την υπομονάδα int2 και την εντολή S8, ή (β) στο γεγονός ότι οι εντολές εκδίδονται πριν γίνει η ανάγνωση των τελούμενων εισόδου, με αποτέλεσμα οι υπομονάδες να καταλαμβάνονται από εντολές που δεν είναι έτοιμες να εκτελεστούν, κάτι που εδώ συμβαίνει με την υπομονάδα int1 και την εντολή S7.

Τη χρονική στιγμή στην οποία η εντολή S5 αποθηκεύει το αποτέλεσμά της στο φάκελο καταχωρητών για τη δεύτερη επανάληψη, δηλαδή στον κύκλο μηχανής 49, οι εντολές που έχουν εκδοθεί και εκκρεμούν είναι οι S3, S5, S6, S7, S8 και S10 της δεύτερης επανάληψης. Οι εντολές S4 και S9 που μεσολαβούν έχουν ολοκληρωθεί. Η τριάδα πινάκων που περιγράφει την κατάσταση δρομολόγησης των εντολών στον κύκλο αυτόν θα είναι η εξής:

Κατάσταση Εντολών				
εντολή	issued	operands read	executed	result written
S3	√	√		
S4	√	√	√	√
S5	√	√	√	
S6	√			
S7	√			
S8	√	√	√	
S9	√	√	√	√
S10	√			

Κατάσταση Υπομονάδων									
υπομονάδα	busy	op	F_i	F_j	F_k	Q_j	Q_k	R_j	R_k
int1	yes	sdc1		$2	$f6		fpadd2	yes	no
int2	yes	addiu	$2	$2				no	
int3	yes	bne		$2	$10	int2		no	yes
fpadd1	no								
fpadd2	yes	add.d	$f6	$f0	$f10	fpmul1	fpmul2	no	no
fpmul1	yes	div.d	$f0	$f2	$f4			no	no
fpmul2	yes	mul.d	$f10	$f8	$f6			no	no

Κατάσταση Καταχωρητών									
καταχωρητής	…	$2	…	$f0	…	$f6	…	$f10	…
FU		int2		fpmul1		fpadd2		fpmul2	

Στον πρώτο πίνακα συμπεριλάβαμε τις εντολές S4 και S9, παρ' ό,τι έχουν ολοκληρωθεί, ενώ δε συμπεριλάβαμε καμία εντολή πριν την S3 και μετά την S10, που είτε έχουν ολοκληρωθεί είτε δεν έχουν ακόμα εκδοθεί. Έτσι, εκτός της εντολής S5 που γνωρίζουμε ότι γράφει το αποτέλεσμά της στο φάκελο καταχωρητών τη χρονική στιγμή που μελετάμε, η εντολή S8 έχει εκδοθεί, έχει διαβάσει τα τελούμενα εισόδου, και έχει εκτελεστεί. Δεν έχει γράψει ακόμα το αποτέλεσμά της στο φάκελο καταχωρητών, εξαιτίας της εξάρτησης τύπου EMA που δεν επιτρέπει την εγγραφή πριν η εντολή S7 διαβάσει τον καταχωρητή $2. Η εντολή S3 είναι η μόνη που εκτελείται, ενώ οι υπόλοιπες εντολές S6, S7 και S10 περιμένουν να διαβάσουν τα τελούμενα εισόδου που χρειάζονται από το φάκελο καταχωρητών.

Οι εντολές του πρώτου πίνακα εκτός των S4 και S9 συμμετέχουν στο δεύτερο πίνακα, όπου δίνεται η κατάσταση των υπομονάδων όπου θα εκτελεστούν, εκτελούνται, ή εκτελέστηκαν και περιμένουν την εγγραφή του αποτελέσματός τους. Για παράδειγμα, η εντολή S6 βρίσκεται στην υπομονάδα fpadd2, όπου περιμένει τις υπομονάδες fpmul1 και fpmul2 να ολοκληρώσουν την εκτέλεσή τους και να αποθηκεύσουν τα αποτελέσματά τους στο φάκελο καταχωρητών, και συγκεκριμένα στους καταχωρητές $f0 και $f10 αντίστοιχα, ώστε να μπορέσει να τα διαβάσει από αυτούς και να εκτελέσει την πράξη της πρόσθεσης. Η εντολή S8 από την άλλη μεριά, βρίσκεται στην υπομονάδα int2 και δεν περιμένει καμία άλλη υπομονάδα. Έτσι, είτε θα εκτελείται, είτε θα έχει ολοκληρώσει την εκτέλεσή της, αλλά δε θα έχει ολοκληρώσει την εγγραφή του αποτελέσματός της στο φάκελο καταχωρητών. Ούτως ή άλλως, η υπομονάδα int2 θα παραμείνει απασχολημένη με την εντολή S8 μέχρι την ολοκλήρωση της εγγραφής του καταχωρητή $2, και δε θα μπορεί μέχρι τότε να εκτελέσει καμία άλλη εντολή.

Από τις εντολές που συμμετέχουν στο δεύτερο πίνακα, μόνο οι S3, S5, S6 και S8 παράγουν κάποιο αποτέλεσμα για το φάκελο καταχωρητών, κάτι που φαίνεται στον τρίτο πίνακα, τον πίνακα κατάστασης καταχωρητών. Έτσι για παράδειγμα, ο καταχωρητής $f0 φαίνεται ότι θα γραφτεί από την υπομονάδα fpmul1. Ας σημειώσουμε ότι δε θα μπορούσε να υπάρχει άλλη υπομονάδα εκτέλεσης που να γράφει τον ίδιο καταχωρητή, μια που καμία άλλη εντολή που να γράφει τον $f0 δε θα μπορούσε να εκδοθεί, πριν γραφτεί ο καταχωρητής από την υπομονάδα fpmul1 και την εντολή S3.

Στην περίπτωση υπερβαθμωτού επεξεργαστή βαθμού 2, ο χρονισμός των εντολών θα γίνει ως εξής:

iter.	instr.	issue	read operands	execute	write result	Σχόλια
1	S1	1	2	3	5	int1
1	S2	1	2	3	6	int2, αναμονή μνήμης
1	S3	2	6	7	27	fpmul1, αναμονή $f2 (AME)
1	S4	2	6	7	9	fpadd1, αναμονή $f2 (AME)
1	S5	3	9	10	18	fpmul2, αναμονή $f8 (AME)
1	S6	5	27	28	30	fpadd2, αναμονή $f6 και $f0
1	S7	5	30	31		int1, αναμονή $f6 (AME)
1	S8	6	7	8	31	int2, αναμονή $2 (EMA)
1	S9	6	7	8	9	int3
1	S10	9	31	32		αναμονή int3, $2 (AME)
2	S1	31	32	33	35	αναμονή $f6 (EME), int2
2	S2	32	33	34	36	αναμονή int1
2	S3	32	36	37	57	fpmul1, αναμονή $f2 (AME)
2	S4	33	36	37	39	fpadd1, αναμονή $f2 (AME)
2	S5	33	39	40	48	fpmul2, αναμονή $f8 (AME)
2	S6	35	57	58	60	fpadd2, αναμονή $f6 και $f0
2	S7	35	60	61		int2, αναμονή $f6 (AME)
2	S8	36	37	38	61	int1, αναμονή $2 (EMA)
2	S9	36	37	38	39	int3
2	S10	39	61	62		αναμονή int3, $2 (AME)
3	S1	61	62	63	65	αναμονή $f6 (EME), int1
3	S2	62	63	64	66	αναμονή int2
3	S3	62	66	67	87	fpmul1, αναμονή $f2 (AME)
3	S4	63	66	67	69	fpadd1, αναμονή $f2 (AME)
3	S5	63	69	70	78	fpmul2, αναμονή $f8 (AME)
3	S6	65	87	88	90	fpadd2, αναμονή $f6 και $f0
3	S7	65	90	91		int1, αναμονή $f6 (AME)
3	S8	66	67	68	91	int2, αναμονή $2 (EMA)
3	S9	66	67	68	69	int3
3	S10	69	91	92		αναμονή int3, $2 (AME)

Όπως και προηγουμένως, οι εξαρτήσεις που δημιουργούν τους μεγαλύτερους κινδύνους στη δρομολόγηση των εντολών είναι η εξάρτηση τύπου EME μεταξύ των εντολών S6 και S1, στον καταχωρητή $f6, και η εξάρτηση τύπου EMA μεταξύ των εντολών S7 και S8, στον καταχωρητή $2. Μια μικρή καθυστέρηση στην έκδοση της S6 είναι αποτέλεσμα του κινδύνου από την εξάρτηση τύπου EME μεταξύ των εντολών S1 και S6, στον καταχωρητή $f6, αλλά η καθυστέρηση αυτή δεν επηρεάζει το μέσο ρυθμό ολοκλήρωσης εντολών.

Τη χρονική στιγμή στην οποία η εντολή S5 αποθηκεύει το αποτέλεσμά της στο φάκελο καταχωρητών για τη δεύτερη επανάληψη, δηλαδή στον 48ο κύκλο μηχανής, οι εντολές που έχουν εκδοθεί και εκκρεμούν είναι οι S3, S5, S6, S7, S8 και S10 της δεύτερης επανάληψης. Οι εντολές S4 και S9 που μεσολαβούν έχουν ολοκληρωθεί. Η τριάδα πινάκων που περιγράφει την κατάσταση δρομολόγησης εντολών στον 48ο κύκλο μηχανής θα είναι η ίδια με αυτήν της προηγούμενης αρχιτεκτονικής που περιέγραφε την κατάσταση δρομολόγησης στον 49ο κύκλο μηχανής, με τη μόνη διαφορά ότι εναλλάσσεται ο ρόλος των υπομονάδων int1 και int2.

Όσο αφορά το ρυθμό ολοκλήρωσης επαναλήψεων του βρόχου, είναι γενικά σκόπιμο να αγνοούμε το χρονισμό της πρώτης επανάληψης, ώστε να αποφεύγουμε λανθασμένες εκτιμήσεις λόγω της αναγκαίας μεταβατικής συμπεριφοράς στην αρχή της εκτέλεσης του βρόχου. Ιδανικά, για ακριβή εκτίμηση, θα πρέπει να αναλύουμε το χρονισμό αρκετών επαναλήψεων, μέχρι να πάρουμε σταθερή συμπεριφορά, και στη συνέχεια να βρίσκουμε το μέσο ρυθμό ολοκλήρωσης μέσα σε έναν αριθμό επαναλήψεων που εμφανίζει περιοδικότητα στο χρονισμό των εντολών. Στη συγκεκριμένη περίπτωση, τέτοια ανάλυση δεν είναι αναγκαία, καθώς γίνεται άμεσα φανερό ότι κάθε επανάληψη απαιτεί 30 κύκλους μηχανής μέχρι την επόμενη, με μέσο ρυθμό ολοκλήρωσης 1 επανάληψη ανά 30 κύκλους μηχανής ή για εντολές 10/30 = 0,333 εντολές ανά κύκλο μηχανής, και για τις δύο αρχιτεκτονικές που μελετήσαμε.

Παρατηρούμε ότι ο μέσος ρυθμός ολοκλήρωσης εντολών – και επαναλήψεων – στην υπερβαθμωτή αρχιτεκτονική είναι ο ίδιος με αυτόν της βαθμωτής, παρά το μεγαλύτερο εύρος της. Μάλιστα, ακόμα κι αν είχαμε επικαλυπτόμενες υπομονάδες κινητής υποδιαστολής, δε θα μπορούσαμε να πάρουμε καλύτερο ρυθμό ολοκλήρωσης, επειδή το άνω φράγμα στο ρυθμό αυτό περιορίζεται από τις εξαρτήσεις τύπου ΕΜΕ και ΕΜΑ που είδαμε.
Γενικά, μπορούμε να συμπεράνουμε ότι το πάγωμα που εφαρμόζεται στην αρχιτεκτονική πίνακα παρακολούθησης για την αντιμετώπιση εξαρτήσεων αυτού του τύπου είναι το πλέον επιζήμιο για την απόδοσή της. Θα πρέπει επομένως να μην επιτρέπουμε την εμφάνιση τέτοιων εξαρτήσεων στον κώδικα, κάτι το οποίο είναι δυνατό, μόνο αν υπάρχει μεγάλος αριθμός καταχωρητών, ώστε ο μεταγλωττιστής να μπορεί να τις αποφύγει, με ένα συνδυασμό ξεδιπλώματος βρόχων και μετονομασίας καταχωρητών.
Το άλλο πρόβλημα της αρχιτεκτονικής αυτής που δεν επιτρέπει υψηλούς ρυθμούς ολοκλήρωσης εντολών είναι η απασχόληση υπομονάδων εκτέλεσης από εντολές που δεν είναι ακόμα έτοιμες να διαβάσουν τα τελούμενα εισόδου τους. Φυσικά, ένας μεγάλος αριθμός υπομονάδων θα περιόριζε το πρόβλημα. Άλλη λύση θα μπορούσε να είναι η ανάγνωση των τελούμενων να γίνεται στον πίνακα παρακολούθησης, πριν την έκδοση στην αντίστοιχη υπομονάδα εκτέλεσης, ή η εισαγωγή ενός αριθμού σταθμών δέσμευσης σε κάθε υπομονάδα εκτέλεσης, κάτι το οποίο σκέφτηκε ο R.Tomasulo, και το οποίο θα μελετήσουμε στην επόμενη άσκηση.

Άσκηση 5:

Έστω ένας υπερβαθμωτός επεξεργαστής βαθμού 2 που εκτελεί εντολές εκτός σειράς με την αρχιτεκτονική των σταθμών δέσμευσης (reservation stations) και τον αλγόριθμο του Tomasulo. Στη ΜΕΔ του επεξεργαστή αυτού περιλαμβάνονται δύο υπομονάδες πρόσθεσης/αφαίρεσης κινητής υποδιαστολής (fpadd) με τρεις σταθμούς δέσμευσης, δύο υπομονάδες πολλαπλασιασμού/ διαίρεσης κινητής υποδιαστολής (fpmul) με τρεις σταθμούς δέσμευσης, και τρεις υπομονάδες ακέραιων λογικών/αριθμητικών πράξεων (int) επίσης με τρεις σταθμούς δέσμευσης. Οι χρόνοι εκτέλεσης των αντίστοιχων εντολών είναι οι εξής:

Εντολή	*Χρόνος εκτέλεσης*
πρόσθεση/αφαίρεση κινητής υποδιαστολής	2cc
πολλαπλασιασμός κινητής υποδιαστολής	8cc
διαίρεση κινητής υποδιαστολής	20cc
ακέραια πράξη	1cc
προσπέλαση μνήμης	+1cc

όπου οι εντολές προσπέλασης μνήμης υπολογίζουν την τελική διεύθυνση προσπέλασης στις υπομονάδες ακέραιων πράξεων, και ολοκληρώνονται σε έναν πρόσθετο κύκλο μηχανής με την προσπέλαση της μνήμης από κατάλληλους σταθμούς φόρτωσης (load buffers) και αποθήκευσης (store buffers). Καμία υπομονάδα δεν υποστηρίζει επικαλυπτόμενες λειτουργίες.
Αν ο επεξεργαστής δέχεται το σύνολο εντολών MIPS, δείξτε σε έναν πίνακα τους χρόνους στους οποίους κάθε εντολή του παρακάτω βρόχου εκδίδεται σε κάποιον σταθμό δέσμευσης, αρχίζει την

εκτέλεσή της, προσπελαύνει τη μνήμη και γράφει το αποτέλεσμά της σε μια αρτηρία CDB, για τις τρεις πρώτες επαναλήψεις του βρόχου, αν ο επεξεργαστής (α) δεν υποστηρίζει, και (β) υποστηρίζει υποθετική εκτέλεση εντολών με τη βοήθεια πίνακα επαναδιάταξης (reorder buffer).

```
loop: ldc1  $f6,32($2)        (S1)
      ldc1  $f2,40($3)        (S2)
      div.d $f0,$f2,$f4       (S3)
      sub.d $f8,$f6,$f2       (S4)
      mul.d $f10,$f8,$f6      (S5)
      add.d $f6,$f0,$f10      (S6)
      sdc1  $f6,32($2)        (S7)
      addiu $2,$2,8           (S8)
      addiu $3,$3,8           (S9)
      bne   $2,$10,loop       (S10)
```

Στην αρχιτεκτονική με υποθετική εκτέλεση να δώσετε και τους χρόνους απόσυρσης των εντολών. Σε κάθε περίπτωση μη ιδανικού χρονισμού των εντολών, να εξηγήσετε το λόγο της καθυστέρησης. Υποθέστε ότι ο φάκελος καταχωρητών είναι ενιαίος με 32 καταχωρητές σταθερής και 32 καταχωρητές κινητής υποδιαστολής, και με όσες θύρες ανάγνωσης/εγγραφής απαιτεί η αρχιτεκτονική για εξυπηρέτηση δύο εντολών ανά κύκλο μηχανής, ότι η μνήμη υποστηρίζει μία προσπέλαση δεδομένων ανά κύκλο μηχανής, με προτεραιότητα στις εντολές φόρτωσης, και ακόμα ότι διατίθενται δύο αρτηρίες CDB για την παροχέτευση των αποτελεσμάτων. Υποθέστε επιτυχημένη πρόβλεψη διακλαδώσεων. Όταν δεν υποστηρίζεται υποθετική εκτέλεση, μια εντολή δε μπορεί να εκτελεστεί, αν προηγείται εντολή διακλάδωσης που εκκρεμεί. Δώστε τους πίνακες δρομολόγησης των εντολών για τον κύκλο μηχανής που η εντολή S5 γράφει το αποτέλεσμά της για τη δεύτερη επανάληψη του βρόχου. Εκτιμήστε το μέσο ρυθμό ολοκλήρωσης των επαναλήψεων του βρόχου, όπως και το μέσο ρυθμό ολοκλήρωσης εντολών, όταν εκτελείται μεγάλος αριθμός επαναλήψεων, και για τις δύο περιπτώσεις αρχιτεκτονικής.

Απάντηση

Σε έναν επεξεργαστή που εκτελεί εντολές εκτός σειράς με τη βοήθεια σταθμών δέσμευσης (reservation stations) και τον αλγόριθμο του R.Tomasulo, χωρίς υποθετική εκτέλεση, κάθε εντολή, αφού αποκωδικοποιηθεί, περνάει διαδοχικά από τις ακόλουθες φάσεις:

1. Έκδοση σε κάποιον σταθμό δέσμευσης. Για να μπορεί μια εντολή να εκδοθεί, αρκεί να υπάρχει κατάλληλος σταθμός δέσμευσης που να είναι διαθέσιμος. Στη φάση αυτή διαβάζονται τα τελούμενα εισόδου που βρίσκονται στο φάκελο καταχωρητών.
2. Εκτέλεση. Η εκτέλεση μιας εντολής μπορεί να ξεκινήσει, μόλις γίνουν διαθέσιμα στο σταθμό δέσμευσης μέσω κάποιας αρτηρίας CDB τα τελούμενα εισόδου που δε βρέθηκαν στο φάκελο καταχωρητών, και εφ' όσον διατίθεται κάποια αντίστοιχη υπομονάδα εκτέλεσης. Αν η εντολή προσπελαύνει τη μνήμη, μεταφέρεται στον κατάλληλο σταθμό φόρτωσης ή αποθήκευσης, όπου και προσπελαύνει τη μνήμη. Ειδικά για εντολές αποθήκευσης, το τελούμενο εισόδου που στέλνεται στη μνήμη μπορεί να παροχετευτεί και στο σταθμό αποθήκευσης, κι έτσι η αντίστοιχη εντολή δε χρειάζεται να περιμένει στο σταθμό ακέραιων πράξεων. Εφ' όσον δεν υποστηρίζεται υποθετική εκτέλεση, μια εντολή δε μπορεί να προχωρήσει στη φάση εκτέλεσης, αν προηγείται διακλάδωση που εκκρεμεί.
3. Εγγραφή αποτελέσματος σε μια αρτηρία CDB. Στη φάση αυτή το αποτέλεσμα μεταφέρεται σε μια από τις διαθέσιμες αρτηρίες CDB, παραλαμβάνεται από τους σταθμούς δέσμευσης που το περιμένουν, και αποθηκεύεται στο φάκελο καταχωρητών.

Για απλούστευση, υποθέτουμε ότι οι παραπάνω φάσεις του κύκλου εντολής – εκτός της φάσης εκτέλεσης – καταλαμβάνουν έναν κύκλο μηχανής. Υποθέτουμε επίσης ότι η προσπέλαση του φακέλου καταχωρητών επιτρέπει εγγραφή στο πρώτο και ανάγνωση στο δεύτερο μισό του κύκλου μηχανής. Η παροχέτευση ενός αποτελέσματος μέσω των αρτηριών CDB γίνεται με είσοδο αυτού σε κάποια αρτηρία στην αρχή ενός κύκλου μηχανής και παραλαβή αυτού από τους αντίστοιχους σταθμούς δέσμευσης μέχρι το τέλος του κύκλου μηχανής, έτσι ώστε η εκτέλεση

στους σταθμούς αυτούς να μπορεί να ξεκινήσει στον επόμενο κύκλο μηχανής, αν δεν περιμένουν άλλο τελούμενο, και εφ' όσον οι απαιτούμενες υπομονάδες εκτέλεσης είναι διαθέσιμες. Ακόμα, υποθέτουμε ότι ένας σταθμός δέσμευσης μπορεί να δεχτεί νέα εντολή στον ίδιο κύκλο μηχανής που απελευθερώνεται από την προηγούμενη, κάτι που συμβαίνει στον κύκλο μηχανής που η τελευταία γράφει το αποτέλεσμά της σε κάποια αρτηρία CDB, ή, σε περίπτωση που αυτή δεν έχει τελούμενο εξόδου, στον κύκλο μηχανής που ολοκληρώνει την εκτέλεσή της. Ειδικά για εντολές προσπέλασης μνήμης, υποθέτουμε ότι η υπομονάδα ακέραιων πράξεων απελευθερώνεται στον κύκλο μηχανής που ολοκληρώνεται ο υπολογισμός της τελικής διεύθυνσης προσπέλασης, η οποία και αντιγράφεται στην αντίστοιχη υπομονάδα προσπέλασης μνήμης. Τέλος, για την υποστήριξη της προτεραιότητας των εντολών φόρτωσης έναντι των εντολών αποθήκευσης στην προσπέλαση μνήμης, υποθέτουμε ότι μια υπομονάδα αποθήκευσης μπορεί να παρέχει τα δεδομένα σε μια εντολή φόρτωσης, όταν υπάρχει επικάλυψη στην τελική διεύθυνση προσπέλασης μεταξύ αυτής και της αντίστοιχης εντολής αποθήκευσης, και η τελευταία προηγείται. Για απλούστευση, θα αγνοήσουμε την πεπερασμένη χωρητικότητα των υπομονάδων προσπέλασης μνήμης.
Όπως και με την περίπτωση της αρχιτεκτονικής του πίνακα παρακολούθησης, έτσι και εδώ, και οι τέσσερις παραπάνω φάσεις του κύκλου εντολής μπορούν να τροποποιηθούν προς διάφορες κατευθύνσεις. Για την παρούσα άσκηση όμως, θα τις χρησιμοποιήσουμε σύμφωνα με το βασικό αλγόριθμο δρομολόγησης εντολών του Tomasulo που περιγράφεται στο βιβλίο των Hennessy και Patterson. Θα δείξουμε δε το ζητούμενο χρονισμό των εντολών σε έναν πίνακα ανά περίπτωση, όπου στις δύο πρώτες στήλες θα έχουμε τον αριθμό επανάληψης (iteration) και τον αριθμό εντολής (instruction) αντίστοιχα, στις επόμενες τέσσερις θα έχουμε τους χρόνους έκδοσης (issue), έναρξης εκτέλεσης (execute), προσπέλασης μνήμης (memory) και αποθήκευσης αποτελέσματος (CDB) αντίστοιχα, ενώ στην τελευταία στήλη θα έχουμε σχόλια που να εξηγούν την καθυστέρηση, αν υπάρχει.

Έτσι, για την αρχιτεκτονική που μελετάμε, υποθέτοντας ότι η πρώτη εντολή εκδίδεται στον κύκλο 1, ο χρονισμός των εντολών θα γίνει ως εξής:

iter.	instr.	issue	execute	memory	CDB	σχόλια
1	S1	1	2	3	4	rsint1, int1
1	S2	1	2	4	5	rsint2, int2, αναμονή μνήμης
1	S3	2	6	-	26	rsmul1, fpmul1, αναμονή $f2 (AME)
1	S4	2	6	-	8	rsadd1, fpadd1, αναμονή $f2 (AME)
1	S5	3	9	-	17	rsmul2, fpmul2, αναμονή $f8 (AME)
1	S6	3	27	-	29	rsadd2, fpadd1, αναμονή $f0 (AME)
1	S7	4	5	30	-	rsint1, int1, αναμονή $f6 (AME)
1	S8	4	5	-	6	rsint2, int2
1	S9	5	6	-	7	rsint1, int1
1	S10	5	7	-	-	rsint3, int1, αναμονή $2 (AME)
2	S1	6	8	9	10	rsint2, int1, αναμονή διακλάδωσης
2	S2	7	8	10	11	int2, αναμονή rsint1, μνήμης
2	S3	7	17	-	37	rsmul3, αναμονή $f2 (AME), fpmul2
2	S4	8	12	-	14	rsadd1, fpadd1, αναμονή $f2 (AME)
2	S5	17	26	-	34	αναμονή rsmul2, fpmul1
2	S6	17	38	-	40	rsadd1, fpadd1, αναμονή $f0 (AME)
2	S7	18	19	41	-	rsint1, int1, αναμονή $f6 (AME)
2	S8	18	19	-	20	rsint2, int2
2	S9	19	20	-	21	rsint1, int1
2	S10	19	21	-	-	rsint3, int1, αναμονή $2 (AME)
3	S1	20	22	23	24	rsint2, int1, αναμονή διακλάδωσης
3	S2	21	22	24	25	int2, αναμονή rsint1, μνήμης

3	S3	26	34	-	54	αναμονή rsmul1, fpmul1
3	S4	26	27	-	29	rsadd3, fpadd2
3	S5	34	37	-	45	αναμονή rsmul2, fpmul2
3	S6	34	55	-	57	rsadd2, fpadd1, αναμονή $f0 (AME)
3	S7	35	36	58	-	rsint1, int1, αναμονή $f6 (AME)
3	S8	35	36	-	37	rsint2, int2
3	S9	36	37	-	38	rsint1, int1
3	S10	36	38	-	-	rsint3, int1, αναμονή $2 (AME)

όπου στην τελευταία στήλη δείχνουμε το συγκεκριμένο σταθμό δέσμευσης στον οποίο εκδίδεται, και τη συγκεκριμένη υπομονάδα στην οποία εκτελείται η εντολή, καθώς και τον όποιο λόγο καθυστέρησης σε μια ή περισσότερες από τις φάσεις του κύκλου εντολής, που μπορεί να είναι κάποια εξάρτηση από δεδομένα, ή δομική εξάρτηση στην έκδοση σε κάποιο σταθμό δέσμευσης ή στην εκτέλεση σε κάποια υπομονάδα, ή στην προσπέλαση της μνήμης.
Παρατηρούμε ότι στις πιο πολλές περιπτώσεις καθυστέρησης από τη δεύτερη επανάληψη και μετά, η αναμονή οφείλεται κυρίως σε δομικές εξαρτήσεις στη φάση έκδοσης ή στη φάση εκτέλεσης. Για παράδειγμα, η εντολή S5 της δεύτερης επανάληψης καθυστερεί τόσο στη φάση έκδοσης, μέχρι κάποιος κατάλληλος σταθμός δέσμευσης – εδώ ο rsmul2 – να γίνει διαθέσιμος, όσο και στη φάση εκτέλεσης, μέχρι κάποια υπομονάδα – εδώ η fpmul1 – να γίνει διαθέσιμη.
Η έλλειψη διαθέσιμων σταθμών δέσμευσης εν μέρει οφείλεται στο γεγονός ότι οι εντολές εκδίδονται πριν γίνει η παραλαβή των τελούμενων εισόδου, με αποτέλεσμα οι σταθμοί δέσμευσης να καταλαμβάνονται από εντολές που δεν είναι έτοιμες να εκτελεστούν. Κάτι τέτοιο συμβαίνει εδώ με την εντολή S6, η οποία καταλαμβάνει ένα σταθμό δέσμευσης τύπου fpadd και στη συνέχεια περιμένει το αποτέλεσμα της υπομονάδας fpmul2 για 21 – στην πρώτη επανάληψη 24 – κύκλους μηχανής.
Επειδή στην αρχιτεκτονική που μελετάμε οι χρόνοι εκτέλεσης των πράξεων πολλαπλασιασμού και διαίρεσης κινητής υποδιαστολής είναι μεγάλοι, απαιτούνται περισσότερες υπομονάδες εκτέλεσης, ώστε να εξαλειφτεί η καθυστέρηση στο χρονισμό των εντολών εξαιτίας μη διαθεσιμότητας υπομονάδων εκτέλεσης. Επίσης, απαιτούνται περισσότεροι σταθμοί δέσμευσης – και μάλιστα περισσότεροι από τις αντίστοιχες υπομονάδες, ώστε να αντιμετωπιστεί η καθυστέρηση που οφείλεται στο γεγονός που αναφέραμε πιο πάνω.
Φυσικά, όποια καθυστέρηση οφείλεται σε κίνδυνο από εξάρτηση τύπου AME, δεν αντιμετωπίζεται, παρά μόνο με το μηχανισμό εκτέλεσης εντολών εκτός σειράς που δεν αφήνει τη ΜΕΔ να παγώσει, όσο υπάρχουν εντολές που να μπορούν να εκτελεστούν. Τέλος, αξίζει να παρατηρήσουμε ότι δεν υπάρχουν κίνδυνοι από εξαρτήσεις τύπου EME ή EMA, λόγω της έμμεσης μετονομασίας καταχωρητών μέσω των σταθμών δέσμευσης.

Τη χρονική στιγμή στην οποία η εντολή S5 γράφει το αποτέλεσμά της σε μια αρτηρία CDB για τη δεύτερη επανάληψη, δηλαδή στον κύκλο μηχανής 34, οι εντολές που έχουν εκδοθεί και εκκρεμούν είναι οι S3, S5, S6 και S7 της δεύτερης, και η S3 της τρίτης επανάληψης. Οι εντολές που μεσολαβούν, όπως και η S4 της τρίτης επανάληψης, έχουν ολοκληρωθεί. Η τριάδα πινάκων που περιγράφει την κατάσταση δρομολόγησης των εντολών στον κύκλο αυτόν θα είναι η εξής:

Κατάσταση Εντολών			
εντολή	issued	executed	CDB written
S3	√		
S4	√	√	√
S5	√	√	
S6	√		
S7	√		
S8	√	√	√
S9	√	√	√

S10	√	√	-
S1	√	√	√
S2	√	√	√
S3	√		
S4	√	√	√

Κατάσταση Σταθμών Δέσμευσης							
σταθμός	busy	op	V_j	V_k	Q_j	Q_k	A
rsint1	no						
rsint2	no						
rsint3	no						
rsadd1	yes	add.d			rsmul3	rsmul2	
rsadd2	no						
rsadd3	no						
rsmul1	yes	div.v	Mem[40+Regs[$3]]	Regs[$f4]			
rsmul2	yes	mul.d	Mem[24+Regs[$2]]-Mem[32+Regs[$3]]	Mem[24+Regs[$2]]			
rsmul3	yes	div.d	Mem[32+Regs[$3]]	Regs[$f4]			
load1	no						
…	no						
store1	yes	sdc1				rsadd1	24+Regs[$2]
…	no						

Κατάσταση Καταχωρητών							
καταχωρητής	…	$f0	…	$f6	…	$f10	…
Q_i		rsmul1		rsadd1		rsmul2	

Στον πρώτο πίνακα συμπεριλάβαμε τις εντολές S4, S8, S9 και S10 της δεύτερης, καθώς και τις εντολές S1, S2 και S4 της τρίτης επανάληψης, παρ' ό,τι έχουν ολοκληρωθεί, ενώ δε συμπεριλάβαμε καμία εντολή πριν την S3 της δεύτερης και μετά την S4 της τρίτης επανάληψης, που είτε έχουν ολοκληρωθεί είτε δεν έχουν ακόμα εκδοθεί. Έτσι, εκτός της εντολής S5 που γνωρίζουμε ότι τη χρονική στιγμή που μελετάμε γράφει το αποτέλεσμά της σε μία από τις δύο αρτηρίες CDB, η εντολή S3 της δεύτερης επανάληψης έχει εκδοθεί στο σταθμό δέσμευσης rsmul3 και εκτελείται στην υπομονάδα fpmul2, η εντολή S6 έχει εκδοθεί στο σταθμό δέσμευσης rsadd1 και περιμένει τα δύο τελούμενα εισόδου της από τους σταθμούς rsmul3 και rsmul2, η εντολή S7 βρίσκεται στο σταθμό αποθήκευσης store1 και περιμένει το δεδομένο αποθήκευσης από το σταθμό rsadd1, ενώ η εντολή S3 της τρίτης επανάληψης έχει εκδοθεί μεν στο σταθμό rsmul1, αλλά δεν έχει υπομονάδα εκτέλεσης για να αρχίσει την εκτέλεσή της.

Οι εντολές του πρώτου πίνακα που δεν έχουν ολοκληρωθεί συμμετέχουν στο δεύτερο πίνακα, όπου δίνεται η κατάσταση των σταθμών δέσμευσης του επεξεργαστή, συμπεριλαμβανομένων των σταθμών φόρτωσης και αποθήκευσης. Για παράδειγμα, η εντολή S6 βρίσκεται στο σταθμό δέσμευσης rsadd1, όπου περιμένει τα δύο τελούμενα εισόδου της, όπως αναφέραμε παραπάνω, τα οποία και θα παραλάβει κάποια στιγμή από τις αρτηρίες CDB, ώστε να μπορέσει να εκτελέσει την πράξη της πρόσθεσης. Η εντολή S7 από την άλλη μεριά, βρίσκεται στο σταθμό store1 και περιμένει το σταθμό rsadd1. Μόλις παραλάβει το δεδομένο προς αποθήκευση, θα προσπελάσει τη μνήμη στη διεύθυνση που βρίσκεται στο πεδίο A του πίνακα. Αν στο μεταξύ εμφανιστεί εντολή φόρτωσης που προσπελαύνει τη μνήμη στην ίδια διεύθυνση, η τελευταία θα παγώσει σε κάποιο σταθμό φόρτωσης, μέχρι την εμφάνιση του δεδομένου που περιμένει ο σταθμός store1, οπότε και θα το παραλάβει από το σταθμό εκείνο.

Από τις εντολές που συμμετέχουν στο δεύτερο πίνακα, μόνο οι S3, S5 και S6 παράγουν κάποιο αποτέλεσμα για το φάκελο καταχωρητών, κάτι που φαίνεται στον τρίτο πίνακα, τον πίνακα κατάστασης καταχωρητών. Έτσι για παράδειγμα, ο καταχωρητής $f0 φαίνεται ότι θα γραφτεί από το σταθμό rsmul1. Παρ' όλο που υπάρχει κι άλλη ενεργή εντολή με τελούμενο εξόδου τον

ίδιο καταχωρητή, ο σταθμός rsmul1 παρέχει την πιο πρόσφατη – κι άρα την έγκυρη – τιμή για τον καταχωρητή $f0.

Στην περίπτωση που ο επεξεργαστής υποστηρίζει υποθετική εκτέλεση με τη βοήθεια πίνακα επαναδιάταξης (reorder buffer), οι φάσεις του κύκλου εντολής τροποποιούνται ως εξής:

1. Έκδοση. Τώρα, κάθε εντολή που εκδίδεται, εισάγεται στον πίνακα επαναδιάταξης. Έτσι, για να μπορέσει μια εντολή να εκδοθεί, πρέπει εκτός από διαθέσιμος σταθμός δέσμευσης, να υπάρχει και ελεύθερη θέση στον πίνακα επαναδιάταξης. Επίσης, επειδή ο πίνακας επαναδιάταξης αποθηκεύει προσωρινά τα αποτελέσματα των εντολών πριν αυτά αποσταλούν στο φάκελο καταχωρητών, για την ανάγνωση των τελούμενων εισόδου κάθε εντολής πρέπει να ελέγχεται και ο πίνακας επαναδιάταξης. Τότε, θα προτιμηθεί ο τελευταίος αντί του φακέλου καταχωρητών, αν υπάρχει ενεργή εντολή που έχει γράψει το αποτέλεσμά της στον πίνακα.
2. Εκτέλεση. Όπως και προηγουμένως, με τη διαφορά ότι η αποθήκευση στη μνήμη μεταφέρεται στην τελευταία φάση, ενώ δεν υπάρχει περιορισμός που να σχετίζεται με τις εντολές διακλάδωσης. Αν η εντολή εμφανίσει ειδική περίπτωση, αυτή αντιμετωπίζεται στην τελευταία φάση.
3. Εγγραφή αποτελέσματος. Τώρα, το αποτέλεσμα στέλνεται στον πίνακα επαναδιάταξης αντί του φακέλου καταχωρητών. Η παροχέτευση γίνεται όπως και πρώτα.
4. Απόσυρση. Η εντολή αποσύρεται, αν δεν υπάρχει προηγούμενη εντολή που να μην έχει αποσυρθεί. Η φάση αυτή φροντίζει για την ενημέρωση του φακέλου καταχωρητών και της μνήμης, καθώς και για την αντιμετώπιση ειδικών περιπτώσεων με τη σειρά έκδοσης των εντολών. Οι σταθμοί αποθήκευσης ενσωματώνονται στον πίνακα επαναδιάταξης.

Για ευκολία, θα αγνοήσουμε το πεπερασμένο μέγεθος του πίνακα επαναδιάταξης.

Ο πίνακας χρονισμού των εντολών θα περιλαμβάνει και μια στήλη με το χρόνο απόσυρσης (commit), και θα είναι ο πιο κάτω:

iter.	instr.	issue	execute	memory	CDB	commit	σχόλια
1	S1	1	2	3	4	5	rsint1, int1
1	S2	1	2	4	5	6	rsint2, int2, αναμονή μνήμης
1	S3	2	6	-	26	27	rsmul1, fpmul1, αναμονή $f2 (AME)
1	S4	2	6	-	8	27	rsadd1, fpadd1, αναμονή $f2 (AME)
1	S5	3	9	-	17	28	rsmul2, fpmul2, αναμονή $f8 (AME)
1	S6	3	27	-	29	30	rsadd2, fpadd1, αναμονή $f0 (AME)
1	S7	4	5	-	-	30	rsint1, int1, αναμονή $f6 (AME)
1	S8	4	5	-	6	31	rsint2, int2
1	S9	5	6	-	7	31	rsint1, int1
1	S10	5	7	-	-	32	rsint3, int1, αναμονή $2 (AME)
2	S1	6	7	8	9	32	rsint2, int2
2	S2	7	8	9	10	33	int1, αναμονή rsint1
2	S3	7	17	-	37	38	rsmul3, αναμονή $f2 (AME), fpmul2
2	S4	8	11	-	13	38	rsadd1, fpadd1, αναμονή $f2 (AME)
2	S5	17	26	-	34	39	αναμονή rsmul2, fpmul1
2	S6	17	38	-	40	41	rsadd1, fpadd1, αναμονή $f0 (AME)
2	S7	18	19	-	-	41	rsint1, int1, αναμονή $f6 (AME)
2	S8	18	19	-	20	42	rsint2, int2
2	S9	19	20	-	21	42	rsint1, int1
2	S10	19	21	-	-	43	rsint3, int1, αναμονή $2 (AME)
3	S1	20	21	22	23	43	rsint2, int2
3	S2	21	22	23	24	44	int1, αναμονή rsint1

3	S3	26	34	-	54	55	αναμονή rsmul1, fpmul1
3	S4	26	27	-	29	55	rsadd3, fpadd2
3	S5	34	37	-	45	56	αναμονή rsmul2, fpmul2
3	S6	34	55	-	57	58	rsadd2, fpadd1, αναμονή $f0 (AME)
3	S7	35	36	-	-	58	rsint1, int1, αναμονή $f6 (AME)
3	S8	35	36	-	37	59	rsint2, int2
3	S9	36	37	-	38	59	rsint1, int1
3	S10	36	38	-	-	60	rsint3, int1, αναμονή $2 (AME)

Είναι φανερό ότι ο χρονισμός των εντολών δε βελτιώθηκε με την υποθετική εκτέλεση, παρόλο που κάποιες εντολές εκδίδονται τώρα νωρίτερα απ' ό,τι στην προηγούμενη αρχιτεκτονική. Το γεγονός αυτό οφείλεται στο ότι οι εξαρτήσεις που αναφέραμε και νωρίτερα δημιουργούν πάλι τους ίδιους κινδύνους, επειδή δε σχετίζονται με το αν ο επεξεργαστής υποστηρίζει ή όχι υποθετική εκτέλεση.

Τη χρονική στιγμή στην οποία η εντολή S5 γράφει το αποτέλεσμά της σε μια αρτηρία CDB για τη δεύτερη επανάληψη, δηλαδή στον κύκλο μηχανής 34, οι εντολές που έχουν εκδοθεί και εκκρεμούν είναι οι S3, S4, S5, S6, S7, S8, S9 και S10 της δεύτερης, και οι S1, S2, S3 και S4 της τρίτης επανάληψης. Οι εντολές που στην ίδια χρονική στιγμή στην προηγούμενη αρχιτεκτονική είχαν ολοκληρωθεί, τώρα περιμένουν στον πίνακα επαναδιάταξης για την απόσυρσή τους. Η τριάδα πινάκων που περιγράφει την κατάσταση δρομολόγησης των εντολών στον κύκλο 34 θα είναι τώρα η εξής:

Πίνακας Επαναδιάταξης					
θέση	busy	instr.	state	dest.	value
1	yes	S3	issued	$f0	
2	yes	S4	CDB written	$f8	Mem[32+Regs[$2]]-Mem[40+Regs[$3]]
3	yes	S5	executed	$f10	
4	yes	S6	issued	$f6	
5	yes	S7	executed	32+Regs[$2]	
6	yes	S8	CDB written	$2	Regs[$2]+8
7	yes	S9	CDB written	$3	Regs[$3]+8
8	yes	S10	CDB written	-	-
9	yes	S1	CDB written	$f6	Mem[40+Regs[$2]]
10	yes	S2	CDB written	$f2	Mem[48+Regs[$2]]
11	yes	S3	issued	$f0	
12	yes	S4	CDB written	$f8	Mem[40+Regs[$2]]-Mem[48+Regs[$3]]
13	no	-	-	-	

Κατάσταση Σταθμών Δέσμευσης								
σταθμός	busy	op	V_j	V_k	Q_j	Q_k	dest	A
rsint1	no							
rsint2	no							
rsint3	no							
rsadd1	yes	add.d			#1	#3	#4, #5	
rsadd2	no							
rsadd3	no							
rsmul1	yes	div.v	Mem[48+Regs[$3]]	Regs[$f4]			#11	
rsmul2	yes	mul.d	Mem[32+Regs[$2]]-Mem[40+Regs[$3]]	Mem[32+Regs[$2]]			#3	
rsmul3	yes	div.d	Mem[40+Regs[$3]]	Regs[$f4]			#1	
load1	no							
…	no							

Κατάσταση Καταχωρητών														
καταχωρητής	…	$2	$3	…	$f0	…	$f2	…	$f6	…	$f8	…	$f10	…
θέση πίνακα	-	#6	#7	-	#11	-	#10	-	#9	-	#12	-	#3	-

Ο πρώτος πίνακας, που είναι ο πίνακας επαναδιάταξης, εμπεριέχει τον πίνακα κατάστασης εντολών που είχαμε στην προηγούμενη αρχιτεκτονική. Στον πίνακα αυτόν συμπεριλάβαμε όλες τις εντολές που αναφέραμε πιο πάνω, ενώ δε συμπεριλάβαμε καμία εντολή πριν την S3 της δεύτερης και μετά την S4 της τρίτης επανάληψης, που είτε έχουν αποσυρθεί είτε δεν έχουν ακόμα εκδοθεί. Έτσι, εκτός της εντολής S5 που γνωρίζουμε ότι τη χρονική στιγμή που μελετάμε γράφει το αποτέλεσμά της σε μία από τις δύο αρτηρίες CDB, η εντολή S3 της δεύτερης επανάληψης έχει εκδοθεί και εκτελείται, η εντολή S6 έχει εκδοθεί και περιμένει τα δύο τελούμενα εισόδου της, η εντολή S7 έχει εκτελεστεί και περιμένει το δεδομένο αποθήκευσης, ενώ η εντολή S3 της τρίτης επανάληψης έχει εκδοθεί μεν, αλλά δεν έχει υπομονάδα εκτέλεσης για να αρχίσει την εκτέλεσή της. Όπως μπορούμε να παρατηρήσουμε, η κατάσταση των πιο πάνω εντολών δε διαφέρει από την αντίστοιχη που είδαμε στην προηγούμενη αρχιτεκτονική. Οι υπόλοιπες εντολές του πίνακα έχουν ολοκληρώσει την εκτέλεσή τους και την εγγραφή του αποτελέσματός τους σε κάποια αρτηρία CDB, αλλά δεν έχουν αποσυρθεί.
Οι εντολές του πρώτου πίνακα που δεν έχουν ολοκληρωθεί συμμετέχουν στο δεύτερο πίνακα, όπου δίνεται η κατάσταση των σταθμών δέσμευσης του επεξεργαστή, συμπεριλαμβανομένων των σταθμών φόρτωσης. Στην αρχιτεκτονική αυτή δεν υπάρχουν σταθμοί αποθήκευσης, και ο πίνακας επαναδιάταξης διατηρεί τα δεδομένα προς αποθήκευση, πριν αυτά εγγραφούν στη μνήμη. Οι σταθμοί δέσμευσης βρίσκονται στην ίδια κατάσταση που βρίσκονταν στην αντίστοιχη χρονική στιγμή της προηγούμενης αρχιτεκτονικής. Αντί των σταθμών δέσμευσης από όπου θα προέλθουν τα δεδομένα που αναμένονται από κάποιες εντολές, δείχνονται οι αντίστοιχες θέσεις του πίνακα επαναδιάταξης, ενώ δείχνονται και οι θέσεις στις οποίες θα αποθηκευτεί το αποτέλεσμα κάθε σταθμού, αν υπάρχει.
Σε αντίθεση όμως με την προηγούμενη αρχιτεκτονική, οι εντολές που έχουν ολοκληρώσει την εκτέλεσή τους και έχουν γράψει το αποτέλεσμά τους σε κάποια αρτηρία CDB, αλλά δεν έχουν ακόμα αποσυρθεί, δεν έχουν εγγράψει το φάκελο καταχωρητών. Γι' αυτό και όλες οι εντολές που βρίσκονται στον πίνακα επαναδιάταξης κι έχουν κάποιο αποτέλεσμα, συμμετέχουν στον τρίτο πίνακα, τον πίνακα κατάστασης καταχωρητών, εκτός αν έχουν εκδοθεί πιο πρόσφατες εντολές που γράφουν τους ίδιους καταχωρητές, οπότε στον πίνακα φαίνονται μόνο οι τελευταίες. Έτσι για παράδειγμα, ο καταχωρητής $f0 θα γραφτεί από τη θέση #11 του πίνακα επαναδιάταξης, παρ' όλο που και η εντολή στη θέση #1 έχει τελούμενο εξόδου τον ίδιο καταχωρητή. Αν μέχρι την απόσυρση της εντολής στη θέση #11 εκδοθεί νέα εντολή με τελούμενο εξόδου τον καταχωρητή $f0, η θέση της νέας αυτής εντολής θα αντικαταστήσει την τιμή που δείχνει ο πίνακας για τον $f0. Με τον τρόπο αυτό, η εγγραφή του καταχωρητή $f0 στο φάκελο καταχωρητών ενδέχεται να γίνει στο τέλος του βρόχου, κάτι το οποίο δε μας ενοχλεί, αφού οι τιμές του καταχωρητή που κάθε φορά μας ενδιαφέρουν διατηρούνται στον πίνακα επαναδιάταξης. Από την άλλη μεριά, ο καταχωρητής $2 θα γραφτεί από τη θέση #6 του πίνακα επαναδιάταξης, που και αυτό μπορεί να αλλάξει, αν πριν την απόσυρση της εντολής S8 που βρίσκεται στη θέση αυτή, εκδοθεί νέα εντολή S8 επόμενης επανάληψης. Παρατηρήστε ότι επειδή ο καταχωρητής $2 δεν έχει ακόμα γραφτεί, οι εκφράσεις των τιμών που δίνονται στους πίνακες και που τον περιέχουν, διαφέρουν από τις αντίστοιχες τις προηγούμενης αρχιτεκτονικής, αλλά στην πραγματικότητα είναι εκφράσεις των ίδιων τιμών.

Θα κλείσουμε την παρούσα άσκηση με μελέτη του ρυθμού ολοκλήρωσης των επαναλήψεων του βρόχου καθώς και του αντίστοιχου ρυθμού ολοκλήρωσης εντολών. Επειδή δεν παρατηρείται περιοδικότητα στο χρονισμό των εντολών σε διαδοχικές επαναλήψεις του βρόχου στο χρόνο που έχουμε αναλύσει για καμία από τις δύο αρχιτεκτονικές, θα χρειαστούμε περαιτέρω ανάλυση του χρονισμού των εντολών. Έτσι, μετά από δυο-τρεις ακόμα επαναλήψεις, βρίσκουμε και για τις δύο αρχιτεκτονικές μια περιοδικότητα που δίνει 28 κύκλους μηχανής ανά 2 επαναλήψεις του βρόχου, επομένως ένα μέσο ρυθμό ολοκλήρωσης επαναλήψεων 1 επανάληψη

ανά 14 κύκλους μηχανής, και ένα μέσο ρυθμό ολοκλήρωσης εντολών 10/14 = 0,714 εντολές ανά κύκλο μηχανής.

Το γεγονός ότι η εκτέλεση των εντολών διακλάδωσης γίνεται πολύ γρήγορα μετά την έκδοσή τους, είναι ένας λόγος που η αρχιτεκτονική με υποθετική εκτέλεση δεν εμφανίζεται να έχει καλύτερη απόδοση από την αρχιτεκτονική χωρίς υποθετική εκτέλεση. Αν η εκτέλεση των εντολών διακλάδωσης καθυστερούσε περισσότερο, τότε η υποθετική εκτέλεση θα μας επέτρεπε μεγαλύτερη επικάλυψη στην εκτέλεση των εντολών, και επομένως υψηλότερους ρυθμούς ολοκλήρωσης.
Συνολικά πάντως, οι ρυθμοί ολοκλήρωσης είναι χαμηλοί σχετικά με το μέγιστο ρυθμό των 2 εντολών ανά κύκλο μηχανής που επιτρέπει η υπερβαθμωτή αρχιτεκτονική. Αυτό οφείλεται στις δομικές εξαρτήσεις που αναφέραμε νωρίτερα, και μάλιστα, αν αναλύσουμε το χρονισμό των υπομονάδων πολλαπλασιασμού/διαίρεσης κινητής υποδιαστολής, θα δούμε ότι η περιορισμένη επικάλυψη που επιτρέπει η ύπαρξη δύο τέτοιων υπομονάδων είναι αυτή που δίνει μέσους χρόνους ολοκλήρωσης μεταξύ διαδοχικών επαναλήψεων κάτω από το χρόνο εκτέλεσης μιας διαίρεσης, της πιο χρονοβόρας δηλαδή πράξης στον κώδικα που έχουμε. Πιο συγκεκριμένα, για μια διαίρεση κι έναν πολλαπλασιασμό κινητής υποδιαστολής απαιτούνται 20+8 = 28 κύκλοι μηχανής, οπότε με δύο υπομονάδες απαιτούνται ως ελάχιστος χρόνος 28/2 = 14 κύκλοι μηχανής ανά επανάληψη, που είναι ακριβώς ο χρόνος που βρήκαμε πιο πάνω. Για επίτευξη υψηλότερων ρυθμών ολοκλήρωσης επαναλήψεων και εντολών απαιτούνται περισσότερες υπομονάδες εκτέλεσης, αλλά και τεχνικές από το μεταγλωττιστή, ώστε να αυξηθεί ο βαθμός διαθέσιμου παραλληλισμού σε επίπεδο εντολών (ILP).

Άσκηση 6:

Έστω ένας μηχανισμός δυναμικής πρόβλεψης διακλαδώσεων δύο επιπέδων που χρησιμοποιείται για την εκτέλεση του πιο κάτω αποσπάσματος κώδικα C:

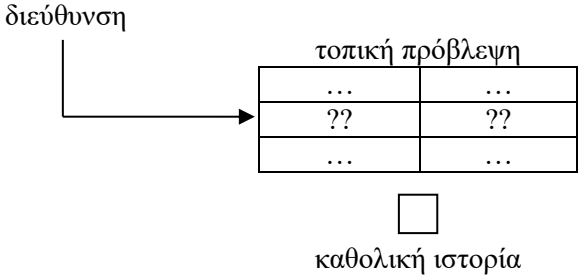
```
if (x == 0) z = 1;
…
if (y == 0) z = 1;
…
if (z == 1) { … }
```

όπου ο κώδικας που υπάρχει δίπλα σε κάθε εντολή διακλάδωσης εκτελείται όταν το άλμα της διακλάδωσης δεν *εκτελείται. Αν μεταξύ των εντολών διακλάδωσης δε μεσολαβούν άλλες διακλαδώσεις, προορισμοί διακλαδώσεων ή αναθέσεις στη μεταβλητή z, δώστε τη μορφή και τις τιμές που θα έχει ο πίνακας ιστορίας διακλαδώσεων για τις τρεις πιο πάνω διακλαδώσεις, υποθέτοντας ότι ο πίνακας είναι τέτοιου μεγέθους, ώστε οι διακλαδώσεις να μην απεικονίζονται στην ίδια γραμμή, (α) για μηχανισμό (1,2) και (β) για μηχανισμό (2,2). Ο κώδικας βρίσκεται στο σώμα ενός βρόχου που έχει ήδη εκτελέσει αρκετές επαναλήψεις, ώστε οι μεταβλητές x και y να έχουν πάρει τουλάχιστον δύο φορές τιμή 0. Πώς αλλάζει η απάντησή σας αν πριν τη δεύτερη εντολή if υπάρχει η λέξη-κλειδί else; Πώς αλλάζει η απάντησή σας και για τις δύο περιπτώσεις, αν γνωρίζετε ότι στην αρχή του παραπάνω κώδικα η μεταβλητή z έχει πάντα τιμή 0;*

Απάντηση

Η καθολική ιστορία παρέχει τη δυνατότητα πρόβλεψης μιας διακλάδωσης με βάση την εκτέλεση προηγούμενων διακλαδώσεων. Με ένα ψηφίο καθολικής ιστορίας μπορούμε για κάθε διακλάδωση να επιλέξουμε από 2 διαφορετικές τοπικές ιστορίες, με βάση την εκτέλεση της αμέσως προηγούμενης εντολής διακλάδωσης. Αν βρισκόμαστε σε ένα βρόχο με μία μόνο διακλάδωση, αυτή η αμέσως προηγούμενη διακλάδωση θα είναι η προηγούμενη εκτέλεση της ίδιας εντολής, διαφορετικά θα είναι άλλη εντολή διακλάδωσης.

Σχηματικά, ο πίνακας ιστορίας διακλαδώσεων με ένα ψηφίο καθολικής ιστορίας θα έχει τη μορφή που δίνεται πιο κάτω. Αν το ψηφίο καθολικής ιστορίας έχει τιμή 0, επιλέγουμε από την



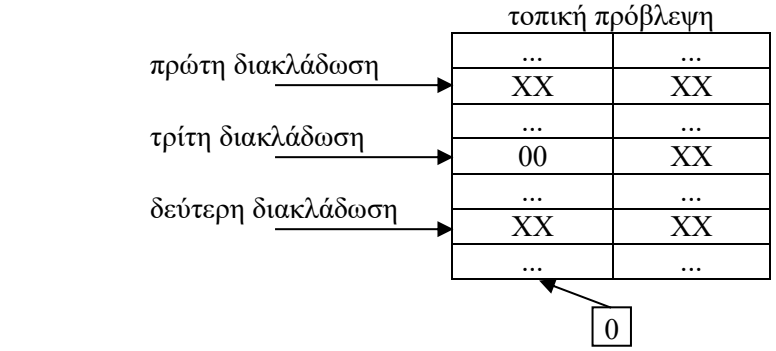
αριστερή στήλη, ενώ αν έχει τιμή 1, επιλέγουμε από τη δεξιά στήλη του πίνακα τοπικής πρόβλεψης.
Όσο αφορά το συγκεκριμένο κώδικα τώρα, κατ' αρχήν παρατηρούμε ότι δεν έχουμε καμία πληροφορία για την εκτέλεση της πρώτης διακλάδωσης, που εξαρτάται από την τιμή της μεταβλητής x. Έτσι, ο πίνακας ιστορίας διακλαδώσεων για την πρώτη διακλάδωση θα περιέχει τιμές που δε μπορούμε να προσδιορίσουμε.
Το ίδιο συμβαίνει και με τη δεύτερη διακλάδωση, οπότε και γι' αυτήν ο πίνακας ιστορίας θα περιέχει απροσδιόριστες τιμές.
Η τρίτη διακλάδωση όμως, κι εφ' όσον δε μεσολαβεί άλλη εντολή διακλάδωσης, είναι συσχετισμένη με τις προηγούμενες, επειδή η συμπεριφορά της εξαρτάται από το αποτέλεσμα της πρώτης ή της δεύτερης. Πιο συγκεκριμένα, μπορούμε να δούμε ότι το άλμα της τρίτης δεν εκτελείται, όταν η μεταβλητή y έχει τιμή 1, κάτι που θα συμβαίνει τουλάχιστον όταν το άλμα της πρώτης διακλάδωσης δεν εκτελείται, ή όταν το άλμα της δεύτερης διακλάδωσης δεν εκτελείται. Οι συσχετίσεις αυτές είναι οι μόνες που αφορούν την τρίτη διακλάδωση, αφού δε μας παρέχεται καμία πληροφορία από τον κώδικα για το πότε η μεταβλητή y έχει τιμή διάφορη του 1.

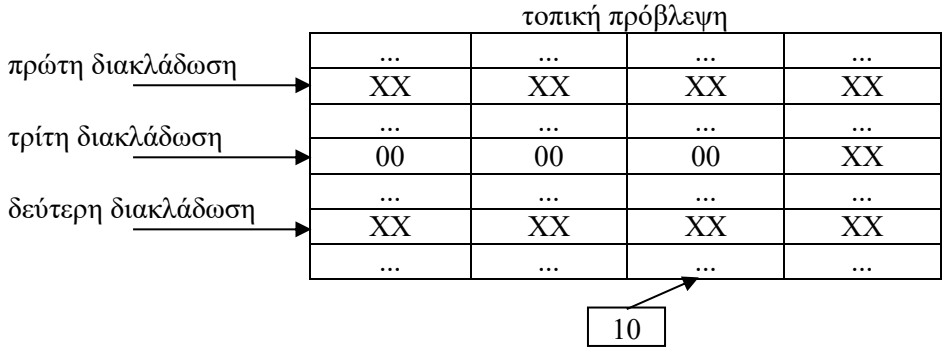
Από τα παραπάνω μπορούμε να συμπεράνουμε ότι όταν είμαστε στην τρίτη διακλάδωση, οπότε το ψηφίο καθολικής ιστορίας αναφέρεται στη δεύτερη, η τοπική πρόβλεψη θα είναι '00', όταν το ψηφίο καθολικής ιστορίας έχει τιμή 0, και απροσδιόριστη, διαφορετικά.
Αναλυτικότερα, η τιμή 0 για το ψηφίο καθολικής ιστορίας σημαίνει μη εκτέλεση του άλματος της δεύτερης διακλάδωσης, γεγονός το οποίο δίνει τιμή 1 στη μεταβλητή z, κι επομένως οδηγεί σε μη εκτέλεση του άλματος της τελευταίας διακλάδωσης. Κάθε φορά επομένως που το άλμα της δεύτερης διακλάδωσης δεν εκτελείται, το άλμα της τρίτης δεν εκτελείται, κι έτσι η κατάσταση που αποθηκεύεται στην αριστερή στήλη του πίνακα τοπικής πρόβλεψης θα είναι η '00', κι αυτό μετά από τουλάχιστον δύο τέτοιες περιπτώσεις, στις οποίες η πρόβλεψη θα έχει εμφανίσει κάποια μεταβατική συμπεριφορά, πιθανά διαφορετική της παραπάνω. Με την κατάσταση αυτή, εξασφαλίζεται η σωστή πρόβλεψη μη εκτέλεσης άλματος για την τρίτη διακλάδωση, κάθε φορά που η δεύτερη δεν εκτελεί το άλμα της.
Από την άλλη μεριά, η τιμή 1 για το ψηφίο καθολικής ιστορίας σημαίνει εκτέλεση του άλματος της δεύτερης διακλάδωσης. Στην περίπτωση αυτή όμως δε μας δίνεται πληροφορία για το ποια τιμή έχει λάβει η μεταβλητή z, κι επομένως δε μπορούμε να ξέρουμε αν η τρίτη διακλάδωση εκτελεί ή όχι το άλμα της. Κάθε φορά που το άλμα της δεύτερης διακλάδωσης εκτελείται, όταν φτάνουμε στην τελευταία διακλάδωση, θα πρέπει να προβλέψουμε με βάση τη δεξιά στήλη του πίνακα τοπικής πρόβλεψης, την οποία και θα ενημερώσουμε στη συνέχεια με βάση το αποτέλεσμα της διακλάδωσης. Με αποτέλεσμα που δε μπορούμε όμως να προσδιορίσουμε, η δεξιά στήλη του πίνακα τοπικής πρόβλεψης θα έχει απροσδιόριστες τιμές των ψηφίων κατάστασης.

Παρατηρήστε ότι το ένα ψηφίο ιστορίας δε μπορεί να εκμεταλλευτεί τη συσχέτιση που έχει η τρίτη διακλάδωση με την πρώτη. Παρόλο που η συσχέτιση είναι η ίδια με αυτή που έχει η τρίτη διακλάδωση με τη δεύτερη, η πληροφορία από την εκτέλεση της πρώτης διακλάδωσης χάνεται με την εκτέλεση της δεύτερης, αφού η καθολική ιστορία τότε θα λάβει τιμή με το αποτέλεσμα της εκτέλεσης αυτής. Αν όμως έχουμε μηχανισμό (2,2), δηλαδή δύο ψηφία καθολικής ιστορίας,

τότε μη εκτέλεση άλματος σε μία από τις δύο πρώτες διακλαδώσεις θα σημαίνει καθολική ιστορία 00, 01 ή 10, οπότε για την τρίτη διακλάδωση στις τρεις από τις τέσσερεις πια στήλες του πίνακα τοπικής πρόβλεψης θα έχουμε αποθηκευμένη την κατάσταση ‘00’, ενώ στην τελευταία που αντιστοιχεί στην καθολική ιστορία 11 θα έχουμε απροσδιόριστη τιμή. Έτσι, στην πρόβλεψη της τρίτης διακλάδωσης θα συμμετέχει τόσο η πρώτη όσο και η δεύτερη διακλάδωση.

Μπορούμε να δούμε την ακριβή μορφή των πινάκων για καθολική ιστορία ενός ψηφίου, τη στιγμή που αυτό είναι 0, και για καθολική ιστορία δύο ψηφίων, τη στιγμή που αυτά είναι 10. Θεωρήσαμε ότι οι διευθύνσεις των διακλαδώσεων είναι τέτοιες, ώστε να απεικονίζονται σε θέσεις του πίνακα με τη σειρά που δείχνεται. Οι στήλες αριθμούνται από αριστερά προς τα δεξιά, με βάση την τιμή καθολικής ιστορίας.





Στη συνέχεια θα δούμε πώς αλλάζει η απάντησή μας αν ο κώδικας έχει τη μορφή:

```
if (x == 0) z = 1;
…
else if (y == 0) z = 1;
…
if (z == 1) { … }
```

Η μορφή αυτή του κώδικα επηρεάζει την πρόβλεψη με βάση την καθολική ιστορία, από τη στιγμή που η δεύτερη διακλάδωση δεν εκτελείται πάντα. Έτσι, όταν η πρώτη διακλάδωση δεν εκτελεί άλμα, η μεταβλητή z λαμβάνει τιμή 1, η δεύτερη διακλάδωση δεν εκτελείται, και επομένως φτάνουμε στην τρίτη διακλάδωση με την πιο πρόσφατη καθολική ιστορία από την πρώτη διακλάδωση και όχι από τη δεύτερη. Μάλιστα, σε αυτή την περίπτωση θα έχουμε εκδήλωση συσχέτισης μεταξύ της πρώτης και της τρίτης διακλάδωσης, και η τρίτη διακλάδωση δεν εκτελεί άλμα. Αντίθετα, αν η πρώτη διακλάδωση εκτελεί άλμα, τότε η δεύτερη διακλάδωση εκτελείται, και ανάλογα με το αν εκτελεί ή όχι άλμα, επηρεάζεται πιθανά η τρίτη διακλάδωση.

Ας δούμε αναλυτικά τις δύο περιπτώσεις μηχανισμού πρόβλεψης:

(α) Για μηχανισμό (1,2), και σύμφωνα με τα παραπάνω, η καθολική ιστορία 0 από την πρώτη διακλάδωση είναι ορατή στην τρίτη διακλάδωση και οδηγεί σε μη εκτέλεση άλματος αυτής. Αν όμως η πρώτη διακλάδωση εκτελέσει άλμα, τότε η καθολική ιστορία που θα φτάσει στην τρίτη θα προέρχεται από τη δεύτερη διακλάδωση. Έτσι, καθολική ιστορία 0 σε τέτοια περίπτωση σημαίνει μη εκτέλεση άλματος της δεύτερης διακλάδωσης, επομένως απόδοση τιμής 1 στη μεταβλητή z και πάλι μη εκτέλεση άλματος στην τρίτη διακλάδωση. Συμπεραίνουμε λοιπόν ότι σε κάθε περίπτωση καθολικής ιστορίας 0, η τρίτη διακλάδωση δεν εκτελεί άλμα, οπότε τα ψηφία κατάστασης της τοπικής ιστορίας της διαμορφώνονται σταδιακά σε '00'.

Από την άλλη πλευρά, η μόνη περίπτωση καθολικής ιστορίας 1 αντιστοιχεί σε εκτέλεση άλματος της δεύτερης διακλάδωσης, εφόσον εκτέλεση άλματος της πρώτης οδηγεί αναγκαστικά σε εκτέλεση της δεύτερης διακλάδωσης, οπότε δεν είναι δυνατό η καθολική ιστορία που φτάνει στην τρίτη διακλάδωση να προκύπτει από την πρώτη. Αυτή όμως η περίπτωση, όπως είδαμε και νωρίτερα, αντιστοιχεί σε μη προβλέψιμη τιμή για τη μεταβλητή z, κι επομένως τα ψηφία κατάστασης της τοπικής ιστορίας της τρίτης διακλάδωσης είναι απροσδιόριστα.

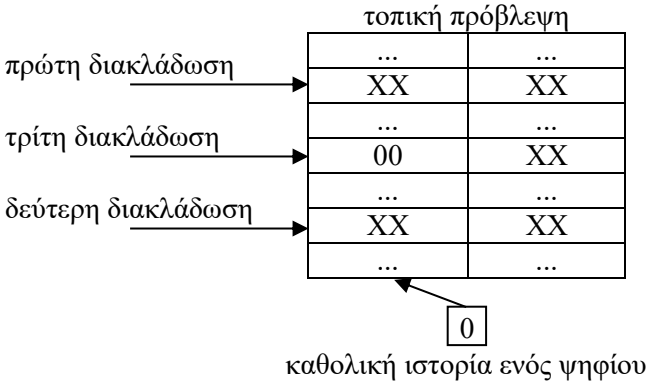
(β) Για μηχανισμό (2,2), καθολική ιστορία 00 αντιστοιχεί στην πρώτη περίπτωση που αναλύσαμε για το μηχανισμό (1,2), δηλαδή σε μη εκτέλεση άλματος στην πρώτη διακλάδωση. Αυτή η τιμή καθολικής ιστορίας δε μπορεί να προκύψει από τη δεύτερη διακλάδωση, γιατί τότε θα έπρεπε το πρώτο ψηφίο να είναι 1 από εκτέλεση άλματος της πρώτης διακλάδωσης! Άρα καθολική ιστορία 00 οδηγεί σίγουρα σε κατάσταση '00' της τρίτης διακλάδωσης.

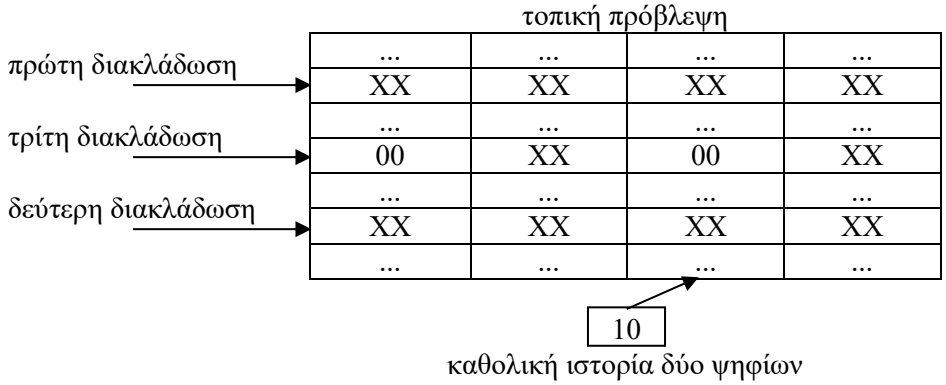
Από την άλλη όμως μεριά, καθολική ιστορία 10 δεν προκύπτει μόνο από εκτέλεση άλματος της πρώτης και μη εκτέλεση άλματος της δεύτερης διακλάδωσης. Σε κάθε περίπτωση που η πρώτη διακλάδωση δεν εκτελεί άλμα, θα υπάρχει κάποια προηγούμενη διακλάδωση που πιθανό να εκτελεί άλμα, οδηγώντας τη ροή κώδικα σε αυτήν. Έτσι, και σε τέτοια περίπτωση μπορούμε να έχουμε καθολική ιστορία 10. Παρατηρούμε όμως ότι είτε έτσι είτε αλλιώς, στην τρίτη διακλάδωση φτάνουμε με τιμή 1 στη μεταβλητή z, κι επομένως αυτή δεν θα εκτελέσει άλμα, και τα ψηφία τοπικής ιστορίας της γίνονται πάλι '00'.

Γνωρίζοντας από παραπάνω ότι εκτέλεση άλματος και στις δύο διακλαδώσεις οδηγεί σε μη απόδοση τιμής στη z, συμπεραίνουμε ότι καθολική ιστορία 11 οδηγεί σε απροσδιόριστη τιμή κατάστασης τοπικής ιστορίας της τρίτης διακλάδωσης.

Η μόνη περίπτωση που παραμένει είναι αυτή της καθολικής ιστορίας 01, η οποία είναι και ιδιαίτερα ενδιαφέρουσα. Παρατηρούμε ότι η πρώτη διακλάδωση είτε δεν εκτελεί άλμα, οπότε φτάνουμε απ' ευθείας στην τρίτη διακλάδωση με *λιγότερο* σημαντικό ψηφίο καθολικής ιστορίας 0, είτε εκτελεί άλμα, οπότε φτάνουμε στην τρίτη διακλάδωση μετά από εκτέλεση και της δεύτερης, άρα με *περισσότερο* σημαντικό ψηφίο καθολικής ιστορίας 1. Τελικά, η περίπτωση καθολικής ιστορίας 01 δεν είναι δυνατό να εμφανιστεί!

Με βάση τα παραπάνω, και εφόσον δεν μπορούμε να κάνουμε καμία εκτίμηση για την εκτέλεση των δύο πρώτων διακλαδώσεων, οι πίνακες ιστορίας για τους δύο μηχανισμούς στην περίπτωση αυτή, για τιμές καθολικής ιστορίας 0 και 10, διαμορφώνονται ως εξής:





Ας εξετάσουμε τώρα και την περίπτωση που της πρώτης εντολής διακλάδωσης προηγείται κάποια εντολή ανάθεσης τιμής 0 στη μεταβλητή z, μέσα στο βρόχο στον οποίο επαναλαμβάνεται ο κώδικας.

Στον αρχικό κώδικα, η τιμή 0 της μεταβλητής z θα φτάσει στην τρίτη διακλάδωση μόνο αν και οι δύο προηγούμενες διακλαδώσεις εκτελέσουν άλμα. Άρα:

(α) Στο μηχανισμό (1,2) δεν αλλάζει κάτι σε σχέση με τα προηγούμενα, από τη στιγμή που το ένα ψηφίο καθολικής ιστορίας δεν αρκεί για να καλυφτεί η περίπτωση εκτέλεσης άλματος και των δύο πρώτων διακλαδώσεων. Καθολική ιστορία 0 οδηγεί σε μη εκτέλεση άλματος της τρίτης διακλάδωσης, ενώ καθολική ιστορία 1 οδηγεί σε απροσδιόριστο αποτέλεσμα της τρίτης, εφόσον δε μπορούμε να γνωρίζουμε τι έκανε η πρώτη διακλάδωση.

(β) Στο μηχανισμό (2,2) όμως, η περίπτωση καθολικής ιστορίας 11 οδηγεί στην τρίτη διακλάδωση με τιμή 0 για τη μεταβλητή z, κι επομένως αυτή εκτελεί άλμα, δίνοντας σταδιακά τιμή ‘11’ στα ψηφία κατάστασης της τοπικής πρόβλεψής της. Παρατηρούμε ότι τώρα η τρίτη διακλάδωση προβλέπεται αποκλειστικά με βάση την καθολική ιστορία!

Και στο δεύτερο κώδικα, η τιμή 0 της μεταβλητής z θα φτάσει στην τρίτη διακλάδωση μόνο αν και οι δύο πρώτες διακλαδώσεις εκτελέσουν άλμα. Η εκτέλεση αυτού του κώδικα διαφέρει από την εκτέλεση του αρχικού μόνο όταν η πρώτη διακλάδωση δεν εκτελεί άλμα, οπότε η δεύτερη δεν εκτελείται. Αυτή όμως η περίπτωση αντιστοιχεί σε απόδοση τιμής 1 στη μεταβλητή z και επομένως δεν μας απασχολεί, αφού την έχουμε ήδη μελετήσει. Όταν τώρα η πρώτη διακλάδωση εκτελεί άλμα, η τιμή 0 της μεταβλητής z διατηρείται μόνο αν και η δεύτερη διακλάδωση εκτελεί άλμα. Επομένως και εδώ τελικά επηρεάζεται μόνο η περίπτωση καθολικής ιστορίας 11, η οποία οδηγεί σε τιμή ‘11’ στα ψηφία κατάστασης της τρίτης διακλάδωσης. Η περίπτωση 01 δεν θα εμφανιστεί, οπότε τα αντίστοιχα ψηφία κατάστασης της τρίτης διακλάδωσης παραμένουν απροσδιόριστα. Η εκτέλεση της τρίτης διακλάδωσης και πάλι προβλέπεται αποκλειστικά με βάση την καθολική ιστορία.

Η σχεδίαση των νέων πινάκων ιστορίας είναι εύκολη και αφήνεται για άσκηση.