ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

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

2η ΑΣΚΗΣΗ ΣΤΗΝ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Ακ. έτος 2007-2008, 5ο Εξάμηνο Σχολή ΗΜ&ΜΥ

Έστω το ακόλουθο τμήμα C κώδικα:

```
for (i = 0; i < 100; i++)
    if (x[i] < a)
        x[i] = a;
    else
        x[i] += b;
```

Θεωρούμε ότι οι τιμές των μεταβλητών a, b βρίσκονται αποθηκευμένες στους καταχωρητές $r3, $r5. Αρχικά, ο καταχωρητής $r2 περιέχει τη διεύθυνση του πρώτου στοιχείου του x, ενώ ο $r6 δείχνει 800 bytes μπροστά. Ο αντίστοιχος κώδικας σε assembly είναι ο εξής:

```
Loop:   lw    $r1, 0($r2)       ; $r1=x[i]
        slt   $r4, $r1, $r3     ; if(x[i]<a) $r4=1 else $r4=0
        beq   $r4, $zero, Else
        sw    $r3, 0($r2)       ; x[i]=a
        j     Cont
Else:   add   $r1, $r1, $r5
        sw    $r1, 0($r2)       ; x[i]+=b
Cont:   add   $r2, $r2, #8
        bne   $r2, $r6, Loop
Exit:
```

α) Υποθέτουμε ότι έχουμε αρχιτεκτονική σωλήνωσης (pipelining) 5 σταδίων (IF ID EX MEM WB). Αρχικά, υποθέτουμε ότι η αρχιτεκτονική σωλήνωσης δε διαθέτει σχήμα προώθησης (forwarding). Επίσης, η εγγραφή σε κάποιο καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή στο δεύτερο μισό του ίδιου κύκλου. Επιπλέον, η ενημέρωση του μετρητή προγράμματος κατά την εκτέλεση μιας εντολής διακλάδωσης γίνεται στο στάδιο MEM και για να γίνει η διακλάδωση πρέπει να εκκενωθεί (flush) το pipeline. Τέλος, στο 75% των περιπτώσεων ισχύει η ανισότητα x[i]< a.
Χρησιμοποιείστε τα κατάλληλα διαγράμματα χρονισμού για να δείξετε την εκτέλεση των εντολών του παραπάνω κώδικα μέσα στην αρχιτεκτονική αγωγού. Υποδείξτε και εξηγείστε τους πιθανούς κινδύνους (hazards) που μπορούν να προκύψουν κατά την εκτέλεση του κώδικα, καθώς και τον τρόπο με τον οποίον αυτοί αντιμετωπίζονται. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του βρόχου;

Το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι TAKEN (x[i]≥a) είναι το ακόλουθο.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lw r1,0(r2)	IF	ID	EX	M	WB																			
slt r4,r1,r3		IF	ID	-	-	EX	M	WB																
beq r4,zero,Else			IF	-	-	ID	-	-	EX	M	WB													
add r1,r1,r5						-	-	-	-	-	IF	ID	EX	M	WB									
sw r1, 0(r2)												IF	ID	-	-	EX	M	WB						
add r2, r2, #8													IF	-	-	ID	EX	M	WB					
bne r2, r6, Loop																IF	ID	-	-	EX	M	WB		
lw r1,0(r2)																	-	-	-	-	-	IF	ID	EX
slt r4,r1,r3																							IF	ID

Στο παραπάνω διάγραμμα, οι "-" υποδηλώνουν stalls. Τα stalls στους κύκλους 4-5 οφείλονται στο ότι ο r1 για την εντολή slt (η ανάγνωση του οποίου γίνεται στο στάδιο ID) γίνεται διαθέσιμος στο τέλος του κύκλου 5 (στάδιο WB). Τα stalls στους κύκλους 7-8 οφείλονται πάλι στο ότι ο r4 για την εντολή beq γίνεται διαθέσιμος στο τέλος του κύκλου 8. Τα stalls στους κύκλους 14-15 οφείλονται στο ότι ο r1 για την εντολή sw γίνεται διαθέσιμος στο τέλος του κύκλου 18. Τα stalls στους κύκλους 18-19 οφείλονται στο ότι ο r2 για την εντολή bne γίνεται διαθέσιμος στο τέλος του κύκλου 19.
Όλα αυτά τα stalls γίνονται προκειμένου να αντιμετωπιστούν οι κίνδυνοι δεδομένων που μπορούν να προκύψουν εξαιτίας RAW εξαρτήσεων μεταξύ των εντολών lw-slt, slt-beq, add-sw και add-bne, αντίστοιχα. Οι κίνδυνοι ελέγχου, που μπορούν να προκύψουν από τις εντολές διακλάδωσης υπό συνθήκη, αντιμετωπίζονται με το να "παγώσουμε" το pipeline μέχρι να γίνει γνωστό το αποτέλεσμα της διακλάδωσης στο στάδιο MEM.

Αντίστοιχα, το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι NOT TAKEN (x[i]<a) είναι αυτό που φαίνεται στη συνέχεια. Τα stalls εξαιτίας των κινδύνων δεδομένων και ελέγχου που πρέπει αντιμετωπιστούν, ακολουθούν την ίδια λογική με πριν.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lw r1,0(r2)	IF	ID	EX	M	WB																			
slt r4,r1,r3		IF	ID	-	-	EX	M	WB																
beq r4,zero,Else			IF	-	-	ID	-	-	EX	M	WB													
sw r3,0(r2)						-	-	-	-	-	IF	ID	EX	M	WB									
j Cont												IF	ID	EX	M	WB								
add r2, r2, #8													-	-	-	IF	ID	EX	M	WB				
bne r2, r6, Loop																	IF	ID	-	-	EX	M	WB	
lw r1,0(r2)																		-	-	-	-	-	IF	ID
slt r4,r1,r3																								IF

Απο τα δύο διαγράμματα χρονισμού συμπεραίνουμε ότι συνολικά, για τις 100 εκτελέσεις του βρόχου, απαιτούνται 100*(25%*21 + 75%*22) + 1 = 2176 κύκλοι.

β) Υποθέτουμε τώρα ότι η αρχιτεκτονική μας διαθέτει όλα τα δυνατά σχήματα προώθησης. Επιπλέον, υποθέτουμε ότι για την πρώτη εντολή διακλάδωσης υπό συνθήκη υπάρχει πρόβλεψη διακλάδωσης που είναι πάντα NOT TAKEN. Στην περίπτωση αυτή, δηλαδή, στο τέλος του σταδίου IF ο μετρητής

προγράμματος θα ανανεώνεται ώστε να δείχνει στην επόμενη σειριακά εντολή. Ο έλεγχος για το αν η πρόβλεψη έγινε σωστά ή όχι γίνεται στο στάδιο MEM. Σε περίπτωση λάθος πρόβλεψης, στο ίδιο στάδιο υπολογίζεται και ο σωστός στόχος της διακλάδωσης, και ανανεώνεται κατάλληλα ο μετρητής προγράμματος. Σε αυτή την περίπτωση, το pipeline πρέπει να εκκενωθεί από τις εντολές που είχαν εισαχθεί σε αυτό εξαιτίας της εσφαλμένης πρόβλεψης.
Όπως και στο προηγούμενο ερώτημα, χρησιμοποιείστε τα κατάλληλα διαγράμματα χρονισμού για να δείξετε την εκτέλεση των εντολών του παραπάνω κώδικα μέσα στην αρχιτεκτονική αγωγού. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του βρόχου;

Το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι TAKEN (x[i]≥a, λάθος πρόβλεψη διακλάδωσης), είναι το ακόλουθο.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lw r1,0(r2)	IF	ID	EX	M	WB																			
slt r4,r1,r3		IF	ID	-	EX	M	WB																	
beq r4,zero,Else			IF	-	ID	EX	M	WB																
add r1,r1,r5					-	-	-	IF	ID	EX	M	WB												
sw r1, 0(r2)									IF	ID	EX	M	WB											
add r2, r2, #8										IF	ID	EX	M	WB										
bne r2, r6, Loop											IF	ID	EX	M	WB									
lw r1,0(r2)												-	-	-	IF	ID	EX	M	WB					
slt r4,r1,r3																IF	ID	-	EX	M	WB			

Όπως είναι άμεσα εμφανές από το διάγραμμα, το σχήμα προώθησης μειώνει τα stalls που είχαμε αρχικά λόγω των κινδύνων δεδομένων. Συγκεκριμένα, στον κύκλο 4 υπάρχει πλέον μόνο ένα stall, και αυτό διότι η τιμή της θέσης μνήμης 0(r2), που πρόκειται να αποθηκευτεί στον r1, δεν μπορεί να είναι διαθέσιμη πριν το στάδιο MEM. Όταν όμως γίνει διαθέσιμη στο τέλος του κύκλου αυτού (και αποθηκευτεί στον ενδιάμεσο καταχωρητή MEM/WB του pipeline), τότε το σχήμα προώθησης θα την προωθήσει κατευθείαν στις εισόδους της ALU, ώστε να μπορεί να εκτελεστεί η εντολή slt r4,r1,r3 χωρίς να χρειάζεται να περιμένουμε μέχρι η τιμή αυτή γραφτεί στον r1. Με αυτόν τον τρόπο, αποφεύγεται το stall που είχαμε στην προηγούμενη περίπτωση στον κύκλο 5.
Ομοίως, τα stalls που είχαμε στους κύκλους 7-8, 14-15 και 18-19, με το σχήμα προώθησης εξαλείφονται. Για παράδειγμα, η τιμή του r4, τον οποίον χρειάζεται η εντολή beq r4,zero,Else στο στάδιο EX, παράγεται στον 5ο κύκλο της εντολής slt r4,r1,r3 κατά το στάδιο EX και αποθηκεύεται στον ενδιάμεσο καταχωρητή EX/MEM. Επομένως μπορεί να προωθηθεί στις εισόδους της ALU και να αποφευχθούν τα stalls. Το ίδιο είδος προώθησης γίνεται και στους κύκλους 14-15 και 18-19.
Όσον αφορά την εντολή διακλάδωσης beq, εφόσον η πρόβλεψη που γίνεται γι'αυτήν είναι πάντα λανθασμένη σε αυτή την περίπτωση, τα stalls που εισάγονται είναι τα ίδια με το ερώτημα α.
(Σημείωση: στην περίπτωση του πρώτου ερωτήματος, όταν εκτελούταν η εντολή beq, το pipeline “πάγωνε”, και εμπόδιζε έτσι τη φόρτωση επόμενων εντολών μέχρι η εκτέλεση της beq να φτάσει στο στάδιο MEM και να γίνει γνωστή η σωστή κατεύθυνση της διακλάδωσης. Στην περίπτωση τώρα που έχουμε πρόβλεψη διακλάδωσης, αλλά αυτή είναι λανθασμένη, θα φορτωθούν οι εντολές της λανθασμένης κατεύθυνσης (sw r3,0(r2), j Cont, κλπ) ενόσω εκτελείται η beq, αλλά μόλις η εκτέλεση της beq φτάσει στο MEM και διαπιστωθεί ότι η πρόβλεψη έχει γίνει λάθος, οι εντολές αυτές θα απορριφθούν, και από τον επόμενο κύκλο θα αρχίζουν να φορτώνονται οι εντολές στη σωστή κατεύθυνση. Τόσο στη μία περίπτωση όσο και στην άλλη, ο χρονισμός του pipeline και τα stalls που εισάγονται τελικά είναι τα ίδια.).

Αντίστοιχα, το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι NOT TAKEN (x[i]<a, σωστή πρόβλεψη διακλάδωσης) είναι αυτό που φαίνεται στη συνέχεια. Οι προωθήσεις που γίνονται στο pipeline ακολουθούν την ίδια λογική με πριν. Επιπλέον, με την πρόβλεψη που γίνεται για την εντολή διακλάδωσης beq, φορτώνεται σωστά η επόμενη στη σειρά εντολή sw από τον 5ο κύκλο, με αποτέλεσμα να

αποφεύγονται τα stalls που είχαμε στην περίπτωση του ερωτήματος α, στην οποία έπρεπε να περιμένουμε μέχρι το στάδιο MEM της beq για να επιλυθεί η διακλάδωση και να φορτωθεί η σωστή επόμενη εντολή.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lw r1,0(r2)	IF	ID	EX	M	WB																			
slt r4,r1,r3		IF	ID	-	EX	M	WB																	
beq r4,zero,Else			IF	-	ID	EX	M	WB																
sw r3,0(r2)					IF	ID	EX	M	WB															
j Cont						IF	ID	EX	M	WB														
add r2, r2, #8							-	-	-	IF	ID	EX	M	WB										
bne r2, r6, Loop											IF	ID	EX	M	WB									
lw r1,0(r2)												-	-	-	IF	ID	EX	M	WB					
slt r4,r1,r3																IF	ID	-	EX	M	WB			

Από τα δύο διαγράμματα χρονισμού συμπεραίνουμε ότι συνολικά απαιτούνται 100*14+1 = 1401 κύκλοι.

γ) Έστω ότι επιθυμούμε την προσθήκη μιας νέας εντολής *φόρτωσης-ενημέρωσης* (lwu), η οποία εκτός από τη συνηθισμένη ανάγνωση από τη μνήμη πραγματοποιεί και ενημέρωση ενός από τους καταχωρητές με τη υπολογιζόμενη διεύθυνση.
Δηλαδή η:

```
lwu   $r3, 1($r4)
```

πραγματοποιεί τις λειτουργίες:

```
$r3 = Mem[$r4 + 1]
$r4 = $r4 + 1;
```

Χωρίς να προσθέσετε επιπλέον write ports στο register file, τι άλλες αλλαγές θα κάνατε στην αρχιτεκτονική αγωγού 5 σταδίων του MIPS για την υλοποίηση της εντολής αυτής; Τι συνέπειες θα είχαν οι όποιες αλλαγές για την υλοποίηση της νέας εντολής (σε throughput, latency και hazards);

Εφόσον δε μπορούμε να χρησιμοποιήσουμε επιπλέον write port στο register file, πρέπει να επεκτείνουμε το pipeline κατά ένα επιπλέον στάδιο για εγγραφή στο αρχείο καταχωρητών, έστω το WB2. Έτσι, για την εντολή lwu r3,1(r4): στο στάδιο EX θα γίνεται η πράξη r4+1, στο στάδιο MEM θα προσπελάζεται το περιεχόμενο της θέσης μνήμης με διεύθυνση r4+1, στο στάδιο WB1 θα γίνεται εγγραφή του περιεχομένου αυτού στον καταχωρητή r3, και στο WB2 θα γίνεται εγγραφή της υπολογισθείσας τιμής r4+1 στον r4. Για όλες τις υπόλοιπες εντολές, στο στάδιο WB2 δε θα επιτελείται καμία λειτουργία.
Έχοντας όμως δύο στάδια στο pipeline στα οποία γίνεται εγγραφή στο αρχείο καταχωρητών, υπάρχει περίπτωση να εμφανιστεί δομικός κίνδυνος όταν μία εντολή lwu ακολουθείται από μια εντολή η οποία γράφει στο αρχείο καταχωρητών, καθώς στον ίδιο κύκλο και οι δύο εντολές θα γράψουν ταυτόχρονα στο αρχείο καταχωρητών, με απρόβλεπτες συνέπειες, π.χ.

lwu r1,0(r2)	IF	ID	EX	MEM	WB1	WB2	
add r4,r4,r8		IF	ID	EX	MEM	WB1	WB2

Γι' αυτό το λόγο, θα πρέπει να προστεθεί λογικό κύκλωμα στο pipeline ώστε όταν ανιχνεύεται η εκτέλεση μιας εντολής lwu, να ελέγχεται πάντα η επόμενή της εντολή. Αν η επόμενη εντολή είναι τέτοια ώστε να γράψει στο αρχείο καταχωρητών στο στάδιο WB1 όταν η lwu γράφει κι αυτή στον ίδιο κύκλο στο WB2, τότε το pipeline θα πρέπει να την καθυστερήσει κατά έναν κύκλο για να αποφευχθεί ο δομικός κίνδυνος, όπως φαίνεται στο παρακάτω διάγραμμα:

lwu r1,0(r2)	IF	ID	EX	MEM	WB1	WB2		
add r4,r4,r8		IF	ID	EX	MEM	-	WB1	WB2

Αν η επόμενη εντολή δε γράφει στο αρχείο καταχωρητών ταυτόχρονα με την lwu (κι αυτό μπορεί να συμβεί είτε αν η εντολή δε γράφει ούτως ή άλλως στο αρχείο καταχωρητών οπότε δεν υπάρχει καμία δραστηριότητα κατά το WB1, ή αν γράφει αλλά έχει καθυστερήσει στο pipeline για άλλους λόγους, π.χ. stalls λόγω κινδύνων δεδομένων), τότε δεν υπάρχει λόγος να καθυστερήσουμε την εκτέλεσή της:

lwu r1,0(r2)	IF	ID	EX	MEM	WB1	WB2		
beq r0,r3,L1		IF	ID	EX	MEM	WB1	WB2	

Μία άλλη τροποποίηση που θα πρέπει να κάνουμε, είναι η επέκταση των ενδιάμεσων καταχωρητών του pipeline με κατάλληλο πεδίο όπου θα αποθηκεύεται η υπολογισθείσα τιμή της διεύθυνσης για την lwu (π.χ. η r4+1 στο αρχικό παράδειγμα) μετά το στάδιο EX, ώστε να αποθηκευτεί στον αντίστοιχο καταχωρητή κατά το στάδιο WB2.

Αυτές οι τροποποιήσεις, εκτός από το επιπλέον stall που εισάγουν για να αποφευχθεί ο δομικός κίνδυνος που περιγράψαμε προηγουμένως, έχουν σαν συνέπεια την αύξηση των stalls στην περίπτωση που πρέπει να αποφευχθεί και κάποιος κίνδυνος δεδομένων, όπως φαίνεται στο ακόλουθο παράδειγμα (έστω ότι δεν έχουμε σχήμα προώθησης):

lwu r1,0(r2)	IF	ID	EX	MEM	WB1	WB2				
beq r2,r0,L1		IF	ID	-	-	-	EX	MEM	WB1	WB2

Σε αυτό το παράδειγμα, η εντολή beq στον 3ο κύκλο χρειάζεται να διαβάσει την τιμή του καταχωρητή r2, που όμως θα γίνει διαθέσιμη στο τέλος του 6ου κύκλου από την εντολή lwu. Έτσι, χρειάζεται να εισάγουμε 3 stalls, δηλαδή ένα παραπάνω σε σχέση με το κλασικό pipeline 5 σταδίων.

Το latency των εντολών, δηλαδή ο αριθμός των κύκλων από τη στιγμή που εισαχθεί μια εντολή στο pipeline μέχρι να βγει από αυτό, στην καλύτερη περίπτωση αυξάνεται κατά 1 κύκλο ρολογιού εξαιτίας του επιπλέον σταδίου.
Το throughput, δηλαδή ο ρυθμός με τον οποίον εξέρχονται οι εντολές από το pipeline, παραμένει -θεωρητικά- ο ίδιος (1 εντολή/κύκλο). Πρακτικά όμως, επειδή με τις νέες τροποποιήσεις όπως είδαμε είναι πιθανό να συμβούν περισσότερα stalls ανά εντολή κατά μέσο όρο, το throughput μειώνεται.

δ) Ξαναγράψτε τον αρχικό κώδικα χρησιμοποιώντας την εντολή lwu. Χρησιμοποιείστε τα κατάλληλα διαγράμματα χρονισμού για να δείξετε την εκτέλεση των εντολών του νέου κώδικα μέσα στην τροποποιημένη σωλήνωση του ερωτήματος γ. Όπως και στο ερώτημα β, υποθέστε ότι η σωλήνωση διαθέτει επιπλέον όλα τα δυνατά σχήματα προώθησης, καθώς και πρόβλεψη διακλάδωσης NOT TAKEN για την πρώτη εντολή διακλάδωσης υπό συνθήκη. Πόσοι κύκλοι απαιτούνται για την εκτέλεση του βρόχου;

Με τη νέα εντολή lwu, και με την προϋπόθεση ότι ο $r2 πριν την εκκίνηση του βρόχου είναι μειωμένος κατά 8 σε σχέση με την αρχική περίπτωση, ο νέος κώδικας μπορεί να γραφτεί ως εξής:

```
Loop:   lwu    $r1, 8($r2)        ;$r1=Mem[r2+8](x[i]), r2=r2+8
        slt    $r4, $r1, $r3      ;if(x[i]<a) $r4=1 else $r4=0
        beq    $r4, $zero, Else
        sw     $r3, 0($r2)        ;Mem[$r2](x[i]) = a
        j      Cont
Else:   add    $r1, $r1, $r5
        sw     $r1, 0($r2)        ;Mem[$r2](x[i]) += b
Cont:   bne    $r2, $r6, Loop
Exit:
```

Το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι TAKEN (x[i]≥a, λάθος πρόβλεψη διακλάδωσης), είναι το ακόλουθο. Σημειώνουμε ότι στην εκτέλεση της εντολής slt, δε χρειάζεται να εισάγουμε επιπλέον stall για την αποφυγή του δομικού κινδύνου με την lwu, καθώς η slt έχει ήδη καθυστερήσει κατά 1 κύκλο λόγω κινδύνου δεδομένων, επομένως η lwu θα βρίσκεται στο WB2, η slt θα βρίσκεται στο MEM και δε θα υπάρχει πρόβλημα.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lwu r1,8(r2)	IF	ID	EX	M	W1	W2																		
slt r4,r1,r3		IF	ID	-	EX	M	W1	W2																
beq r4,zero,Else			IF	-	ID	EX	M	W1	W2															
add r1,r1,r5					-	-	-	IF	ID	EX	M	W1	W2											
sw r1, 0(r2)									IF	ID	EX	M	W1	W2										
bne r2, r6, Loop										IF	ID	EX	M	W1	W2									
lwu r1,8(r2)											-	-	-	IF	ID	EX	M	W1	W2					
slt r4,r1,r3															IF	ID	-	EX	M	W1	W2			

Αντίστοιχα, το διάγραμμα χρονισμού του pipeline για την περίπτωση όπου η εντολή beq είναι NOT TAKEN (x[i]<a, σωστή πρόβλεψη διακλάδωσης) είναι αυτό που φαίνεται στη συνέχεια.

Κύκλος	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
lwu r1,8(r2)	IF	ID	EX	M	W1	W2																		
slt r4,r1,r3		IF	ID	-	EX	M	W1	W2																
beq r4,zero,Else			IF	-	ID	EX	M	W1	W2															
sw r3,0(r2)					IF	ID	EX	M	W1	W2														
j Cont						IF	ID	EX	M	W1	W2													
bne r2, r6, Loop							-	-	-	IF	ID	EX	M	W1	W2									
lwu r1,8(r2)											-	-	-	IF	ID	EX	M	W1	W2					
slt r4,r1,r3															IF	ID	-	EX	M	W1	W2			

Από τα δύο διαγράμματα χρονισμού συμπεραίνουμε ότι συνολικά απαιτούνται 100*13+2 = 1302 κύκλοι.