Pipelines

Γεώργιος Κυριακόπουλος – el18153

1. Το διάγραμμα χρονισμού της περίπτωσης όπου έχουμε αρχιτεκτονική σωλήνωσης χωρίς σχήμα προώθησης φαίνεται παρακάτω. Σε αυτήν την αρχιτεκτονική του MIPS μας ενδιαφέρει να αντιμετωπίσουμε τα true dependencies μεταξύ των δεδομένων, δηλαδή τις RAW (Read-After-Write) εξαρτήσεις που προκύπτουν αρκετές φορές, συνήθως με διπλό stall. Συγκεκριμένα, έχουμε τις εξής RAW εξαρτήσεις:

a) $t1 μεταξύ του read στη LW $t2, 200($t1) και του write στην ADDI $t1, $t1, 4
b) $t2 μεταξύ του read στη LW $t3, 0($t2) και του write στη LW $t2, 200($t1)
c) $t3 μεταξύ του read στην ADD $t2, $t2, $t3 και του write στη LW $t3, 0($t2)
d) $t2 μεταξύ του read στη LW $t4, 100($t2) και του write στην ADD $t2, $t2, $t3
e) $t4 μεταξύ του read στην ADD $t3, $t3, $t4 και του write στη LW $t4, 100($t2)
f) $t3 μεταξύ του read στην ADD $t3, $t3, $t2 και του write στην ADD $t3, $t3, $t4
g) $t3 μεταξύ του read στη SW $t3, 200($t1) και του write στην ADD $t3, $t3, $t2
h) $t9 μεταξύ του read στη BNEZ $t9, LOOP και του write στην ADDI $t9, $t9, -4

Και οι 8 αυτές εξαρτήσεις αντιμετωπίζονται με διπλό stall, ώστε να είμαστε σίγουροι πως οι εντολές που γράφουν τους καταχωρητές φτάνουν στο WB στάδιο και γράφουν στο πρώτο μισό του δεδομένου κύκλου, ώστε στο δεύτερο μισό του κύκλου να διαβάσουν, αντίστοιχα, οι επόμενες εντολές από τους καταχωρητές τα arguments που θέλουν, στο στάδιο ID, μετά από τα δύο stall.

Παρατηρούμε πώς η πρώτη εντολή της δεύτερης επανάληψης γίνεται στον κύκλο 29, καθώς τα branch επιλύονται στο στάδιο EX. Επομένως κάθε επανάληψη απαιτεί 28 κύκλους για να ολοκληρωθεί, εξαιρουμένης της τελευταίας επανάληψης που απαιτεί 30 κύκλους μέχρι και το WB στάδιο της τελευταίας (BNEZ $t9, LOOP) εντολής. Στον καταχωρητή $t9 έχουμε αρχικά την τιμή $0x300 = 768_{10}$ και σε κάθε επανάληψη μειώνεται κατά 4 (ADDI $t9, $t9, -4). Άρα έχουμε συνολικά $768/4 = 192$ επαναλήψεις. Άρα ο συνολικός αριθμός κύκλων που απαιτείται για να ολοκληρωθεί ο δοσμένος βρόχος είναι ίσος με $191 * 28 + 30$ ή $192 * 28 + 2 = 5378$.

CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ADDI $t1, $t1, 4	IF	ID	EX	ME	WB										
LW $t2, 200($t1)		IF	ID	-	-	EX	ME	WB							
LW $t3, 0($t2)			IF	-	-	ID	-	-	EX	ME	WB				
ADD $t2, $t2, $t3				-	-	IF	-	-	ID	-	-	EX	ME	WB	
LW $t4, 100($t2)				-	-		-	-	IF	-	-	ID	-	-	EX
ADD $t3, $t3, $t4				-	-		-	-		-	-	IF	-	-	ID
ADD $t3, $t3, $t2				-	-		-	-		-	-		-	-	IF
SW $t3, 200($t1)				-	-		-	-		-	-		-	-	
ADDI $t9, $t9, -4				-	-		-	-		-	-		-	-	
BNEZ $t9, LOOP				-	-		-	-		-	-		-	-	

CC	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
LW $t4, 100($t2)	ME	WB													
ADD $t3, $t3, $t4	-	-	EX	ME	WB										
ADD $t3, $t3, $t2	-	-	ID	-	-	EX	ME	WB							
SW $t3, 200($t1)	-	-	IF	-	-	ID	-	-	EX	ME	WB				
ADDI $t9, $t9, -4	-	-		-	-	IF	-	-	ID	EX	ME	WB			
BNEZ $t9, LOOP	-	-		-	-		-	-	IF	ID	-	-	EX	ME	WB
ADDI $t1, $t1, 4	-	-		-	-		-	-			-	-		IF	ID

2. Το νέο διάγραμμα χρονισμού για την αρχιτεκτονική με τα σχήματα προώθησης φαίνεται παρακάτω. Αυτή τη φορά έχουμε καταφέρει για τις εξαρτήσεις RAW a, d, f, g, h να αφαιρέσουμε τελείως τα δύο stall που είχαμε προσθέσει το πρώτο ερώτημα, καθώς σε αυτές τις εξαρτήσεις το αποτέλεσμα της εγγραφής είναι διαθέσιμο στο τέλος του σταδίου EX και με μία προώθηση μπορεί να πάει στο στάδιο EX της επόμενης εντολής που το διαβάζει. Αντιθέτως, για τις εξαρτήσεις b, c, e καταφέραμε να αφαιρέσουμε το ένα stall, διότι το αποτέλεσμα της εγγραφής θα είναι διαθέσιμο στο τέλος του σταδίου ΜΕ (MEM) και με ένα stall και μία προώθηση μπορεί να πάει και αυτό στο στάδιο EX της επόμενης εντολής που το διαβάζει.

Σε αυτήν την αρχιτεκτονική με τις προωθήσεις παρατηρούμε ότι η πρώτη εντολή του δεύτερου κύκλου γίνεται στον κύκλο 16, καθώς και πάλι, τα branch επιλύονται στο στάδιο EX. Επομένως κάθε επανάληψη απαιτεί 15 κύκλους για να ολοκληρωθεί, εξαιρουμένης της τελευταίας που απαιτεί 17 κύκλους μέχρι και το WB στάδιο της τελευταίας (BNEZ $t9, LOOP) εντολής της. Και πάλι, στον καταχωρητή $t9 έχουμε αρχικά την τιμή $0x300 = 768_{10}$ και σε κάθε επανάληψη μειώνεται κατά 4 (ADDI $t9, $t9, -4). Άρα έχουμε και πάλι, συνολικά $768/4 = 192$ επαναλήψεις. Άρα ο συνολικός αριθμός κύκλων που απαιτείται για να ολοκληρωθεί ο δοσμένος βρόχος είναι ίσος με $191 * 15 + 17$ ή $192 * 15 + 2 = 2882$.

CC	1	2	3	4	5	6	7	8	9
ADDI $t1, $t1, 4	IF	ID	EX	ME	WB				
LW $t2, 200($t1)		IF	ID	EX	ME	WB			
LW $t3, 0($t2)			IF	ID	-	EX	ME	WB	
ADD $t2, $t2, $t3				IF	-	ID	-	EX	ME
LW $t4, 100($t2)					-	IF	-	ID	EX
ADD $t3, $t3, $t4					-		-	IF	ID
ADD $t2, $t2, $t3					-		-		IF

CC	10	11	12	13	14	15	16	17
ADD $t2, $t2, $t3	WB							
LW $t4, 100($t2)	ME	WB						
ADD $t3, $t3, $t4	-	EX	ME	WB				
ADD $t3, $t3, $t2	-	ID	EX	ME	WB			
SW $t3, 200($t1)	-	IF	ID	EX	ME	WB		
ADDI $t9, $t9, -4	-		IF	ID	EX	ME	WB	
BNEZ $t9, LOOP	-			IF	ID	EX	ME	WB
ADDI $t1, $t1, 4	-						IF	ID

3. Για να λύσουμε τα 3 stall που έχουμε ακόμα, παρά τη χρήση των σχημάτων προώθησης και να πετύχουμε καλύτερη επίδοση θα αναδιατάξουμε και θα τροποποιήσουμε μερικά σημεία του κώδικα ως έχει:

```
LOOP:  LW $t2, 204($t1)
       ADDI $t1, $t1, 4
       LW $t3, 0($t2)
       ADDI $t9, $t9, -4
       ADD $t2, $t2, $t3
       LW $t4, 100($t2)
       ADD $t3, $t3, $t2
       ADD $t3, $t3, $t4
       SW $t3, 200($t1)
       BNEZ $t9, LOOP
```

Αρχικά, αλλάξαμε σειρά στις δύο πρώτες εντολές, με την απαραίτητη αλλαγή του offset στην LW, διορθώνοντας έτσι το πρώτο stall που περίμενε την εγγραφή του $t2 στην LW $t2, 200($t1) για να το διαβάσει στην LW $t3, 0($t2). Στη συνέχεια, για να διορθώσουμε και το δεύτερο stall βάλαμε την εντολή ADDI $t9, $t9, -4 στο pipeline μεταξύ των δύο εντολών που παρήγαγαν ένα ακόμα RAW hazard, δηλαδή της LW $t3, 0($t2) που έγγραφε στον $t3 και της ADD $t2, $t2, $t3 που διάβαζε τον $t3, καθώς η ADDI είναι ανεξάρτητη από το καταχωρητή ενδιαφέροντος $t3. Τέλος, αλλάξαμε σειρά στις ADD $t3, $t3, $t4 και ADD $t3, $t3, $t2, ώστε να διορθώσουμε και το τρίτο και τελευταίο stall, όπου η ADD $t3, $t3, $t4 περίμενε την LW $t4, 100($t2) να γράψει στον $t4, ώστε να τον διαβάσει αυτή στη συνέχεια. Με την αλλαγή αυτή δεν αλλοιώνεται το περιεχόμενο του κώδικα, καθώς ο συνδυασμός των δύο αυτών εντολών εκτελεί την πράξη $t3 = t3 + t4 + t2$.

Σε αυτήν την αρχιτεκτονική με τον επεξεργασμένο κώδικα παρατηρούμε ότι η πρώτη εντολή του δεύτερου κύκλου γίνεται στον κύκλο 13, καθώς και πάλι, τα branch επιλύονται στο στάδιο EX. Επομένως κάθε επανάληψη απαιτεί 12 κύκλους για να ολοκληρωθεί, εξαιρουμένης της τελευταίας που απαιτεί 14 κύκλους μέχρι και το WB στάδιο της τελευταίας (BNEZ $t9, LOOP) εντολής της. Και πάλι, στον καταχωρητή $t9 έχουμε αρχικά την τιμή $0x300 = 768_{10}$ και σε κάθε επανάληψη μειώνεται κατά 4 (ADDI $t9, $t9, -4). Άρα έχουμε και πάλι, συνολικά $768/4 = 192$ επαναλήψεις. Άρα ο συνολικός αριθμός κύκλων που απαιτείται για να ολοκληρωθεί ο δοσμένος βρόχος είναι ίσος με $191 * 12 + 14$ ή $192 * 12 + 2 = 2306$.

CC	1	2	3	4	5	6	7	8
LW $t2, 204($t1)	IF	ID	EX	ME	WB			
ADDI $t1, $t1, 4		IF	ID	EX	ME	WB		
LW $t3, 0($t2)			IF	ID	EX	ME	WB	
ADDI $t9, $t9, -4				IF	ID	EX	ME	WB
ADD $t2, $t2, $t3					IF	ID	EX	ME
LW $t4, 100($t2)						IF	ID	EX
ADD $t3, $t3, $t2							IF	ID
ADD $t3, $t3, $t4								IF

CC	9	10	11	12	13	14
ADD $t2, $t2, $t3	WB					
LW $t4, 100($t2)	ME	WB				
ADD $t3, $t3, $t2	EX	ME	WB			
ADD $t3, $t3, $t4	ID	EX	ME	WB		
SW $t3, 200($t1)	IF	ID	EX	ME	WB	
BNEZ $t9, LOOP		IF	ID	EX	ME	WB
LW $t2, 204($t1)					IF	ID

Πλέον, με τον ανωτέρω διαμορφωμένο κώδικα έχουμε τις εξής προωθήσεις που λύνουν αντίστοιχα όποια RAW hazards προκύπτουν (καθώς δεν μας ενδιαφέρουν τα WAR ή WAW στην αρχιτεκτονική MIPS):

a) ME(4)→EX(5), όπου προωθούμε το $t2 που γράφεται από τη LW $t2, 204($t1) στη LW $t3, 0($t2) που θέλει να το διαβάσει

b) ME(6)→EX(7), όπου προωθούμε το $t3 που γράφεται από τη LW $t3, 0($t2) στην ADD $t2, $t2, $t3 που θέλει να το διαβάσει

c) EX(7)→EX(8), όπου προωθούμε το $t2 που γράφεται από την ADD $t2, $t2, $t3 στη LW $t4, 100($t2) που θέλει να το διαβάσει

d) ME(9)→EX(10), όπου προωθούμε το $t4 που γράφεται από τη LW $t4, 100($t2) στην ADD $t3, $t3, $t4 που θέλει να το διαβάσει

e) EX(9)→EX(10), όπου προωθούμε το $t3 που γράφεται από την ADD $t3, $t3, $t2 στην ADD $t3, $t3, $t4 που θέλει να το διαβάσει

f) EX(10)→EX(11), όπου προωθούμε το $t3 που γράφεται από την ADD $t3, $t3, $t4 στη SW $t3, 200($t1) που θέλει να το διαβάσει.

Με αυτές τις προωθήσεις, επιτυγχάνουμε το βέλτιστο αποτέλεσμα, όπου δεν έχουμε κανένα stall στο pipeline μας και έχουμε τον ελάχιστο δυνατό αριθμό κύκλων που απαιτεί ο βρόγχος για να ολοκληρωθεί.