2η Άσκηση στην Αρχιτεκτονική Υπολογιστών

Στεφανάκης Γεώργιος (el18436)

Ακ. έτος 2020 – 2021, 5ο Εξάμηνο, Σχολή ΗΜΜΥ

1)

```
 1. LOOP:  ADDI $t1, $t1, 4
 2.        LW   $t2, 200($t1)
 3.        LW   $t3, 0($t2)
 4.        ADD  $t2, $t2, $t3
 5.        LW   $t4, 100($t2)
 6.        ADD  $t3, $t3, $t4
 7.        ADD  $t3, $t3, $t2
 8.        SW   $t3, 200($t1)
 9.        ADDI $t9, $t9, -4
10.        BNEZ $t9, LOOP
```

Παρακάτω φαίνεται το διάγραμμα χρονισμού του κώδικα που δίνεται από την εκφώνηση χωρίς σχήμα προώθησης.

CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
ADDI	IF	ID	EX	MEM	WB																												
LW		IF	ID	-	-	EX	MEM	WB																										
LW			IF	-	-	ID	-	-	EX	MEM	WB																						
ADD						IF	-	-	ID	-	-	EX	MEM	WB																			
LW									IF	-	-	ID	-	-	EX	MEM	WB															
ADD												IF	-	-	ID	-	-	EX	MEM	WB												
ADD															IF	-	-	ID	-	-	EX	MEM	WB									
SW																		IF	-	-	ID	-	-	EX	MEM	WB						
ADDI																					IF	-	-	ID	EX	MEM	WB					
BNEZ																								IF	ID	-	-	EX	MEM	WB			
ADDI																													IF	ID	EX	MEM	WB

- Στις οδηγίες υπ' αριθμόν 1 – 2 έχουμε READ AFTER WRITE (RAW) hazard ως προς τον καταχωρητή $t1
- Στις οδηγίες 2 – 3 έχουμε RAW hazard ως προς τον $t2
- Στις οδηγίες 3 – 4 έχουμε RAW hazard ως προς τον $t3
- Στις οδηγίες 4 – 5 έχουμε RAW hazard ως προς τον $t2
- Στις οδηγίες 5 – 6 έχουμε RAW hazard ως προς τον $t4
- Στις οδηγίες 6 – 7 έχουμε RAW hazard ως προς τον $t3
- Στις οδηγίες 7 – 8 έχουμε RAW hazard ως προς τον $t3
- Στις οδηγίες 9 – 10 έχουμε RAW hazard ως προς τον $t9

Τα παραπάνω hazards αντιμετωπίζονται με stalling για 2 κύκλους μέχρι το στάδιο WB όπου παράγεται η τιμή που είναι απαραίτητη για την εκτέλεση της επόμενης εντολής. Επιπλέον, παρατηρούμε ότι η αρχική τιμή του $t9 είναι σε δεκαδικό σύστημα ίση με $3 \cdot 16^2 = 768$ και σε κάθε επανάληψη αφαιρούμε από αυτήν 4 μονάδες, δηλαδή οι συνολικές επαναλήψεις θα είναι $\frac{768}{4} = 192$. Οι πρώτες 191 επαναλήψεις

απαιτούν 28 κύκλους και η τελευταία απαιτεί 28 + 2, άρα συνολικά απαιτούνται 5378 κύκλοι.

2)

CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ADDI	IF	ID	EX	MEM	WB															
LW		IF	ID	EX	MEM	WB														
LW			IF	ID	-	EX	MEM	WB												
ADD				IF	-	ID	-	EX	MEM	WB										
LW						IF	-	ID	EX	MEM	WB									
ADD								IF	ID	-	EX	MEM	WB							
ADD									IF	-	ID	EX	MEM	WB						
SW											IF	ID	EX	MEM	WB					
ADDI												IF	ID	EX	MEM	WB				
BNEZ													IF	ID	EX	MEM	WB			
ADDI																IF	ID	EX	MEM	WB

Στο παραπάνω διάγραμμα χρονισμού έχουμε προωθήσεις που φαίνονται με τα μπλε βελάκια. Οι προωθήσεις από R-Type εντολές γίνονται από το στάδιο EX, καθώς το αποτέλεσμα είναι διαθέσιμο μετά από τις πράξεις που γίνονται στο κύκλωμα της ALU, και οι προωθήσεις που γίνονται από οδηγίες load και store γίνονται από το στάδιο MEM αφού απαιτείται προσπέλαση της μνήμης. Πλέον ο συνολικός αριθμός κύκλων δίνεται από τη σχέση $191 \cdot 15 + 17 = 2882$.

3) Μπορούμε αρχικά να εναλλάξουμε τις εντολές 1 και 2 και να γλιτώσουμε το stall στον κύκλο 5 προωθώντας την έξοδο του MEM σταδίου της εντολής LW (πλέον εντολή υπ' αριθμόν 1) στην είσοδο του σταδίου EX της εντολής LW (πλέον εντολή υπ' αριθμόν 3).

```
1. LOOP:  LW   $t2, 204($t1)
2.        ADDI $t1, $t1, 4
3.        LW   $t3, 0($t2)
```

Παρατηρούμε ότι έχουμε αλλάξει το offset στην εντολή 1 καθώς στον προηγούμενο κώδικα γινόταν load στον $t2 το κελί μνήμης με διεύθυνση 200 + $t1 αφού είχαμε αυξήσει τον $t1 κατά 4. Μπορούμε επίσης να μεταφέρουμε την εντολή 9 στη θέση 4 καθώς η μεταβλητή $t9 δεν παρουσιάζει RAW hazard σε κανένα σημείο του κώδικα με αποτέλεσμα να γλιτώσουμε και το stall που υπάρχει στον κύκλο 7.

```
1. LOOP:  LW   $t2, 204($t1)
2.        ADDI $t1, $t1, 4
3.        LW   $t3, 0($t2)
4.        ADDI $t9, $t9, -4
```

Η οδηγία 4 που πλέον βρίσκεται στη θέση 5 μπορεί να πάρει την έξοδο του MEM της οδηγίας LW, που είναι πλέον η τρίτη, ως είσοδο στο EX στάδιό της.

Τέλος, μπορούμε να εναλλάξουμε τις εντολές 6 και 7 έτσι ώστε να αποτρέψουμε και το stall στον κύκλο 10. Με αυτό τον τρόπο, μπορούμε να περάσουμε ως είσοδο στο EX στάδιο της 8 την έξοδο του MEM σταδίου της 6 ($t4) και την έξοδο του EX σταδίου της 7 ($t3).

```
1. LOOP:  LW   $t2, 204($t1)
2.        ADDI $t1, $t1, 4
3.        LW   $t3, 0($t2)
4.        ADDI $t9, $t9, -4
5.        ADD  $t2, $t2, $t3
6.        LW   $t4, 100($t2)
7.        ADD  $t3, $t3, $t2
8.        ADD  $t3, $t3, $t4
9.        SW   $t3, 200($t1)
10.       BNEZ $t9, LOOP
```

Με αυτές τις αλλαγές καταφέραμε να απαλείψουμε όλα τα stalls που προκύπτανε από τον αρχικό κώδικα. Το διάγραμμα χρονισμού παρακάτω.

CC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
LW	IF	ID	EX	MEM	WB												
ADDI		IF	ID	EX	MEM	WB											
LW			IF	ID	EX	MEM	WB										
ADDI				IF	ID	EX	MEM	WB									
ADD					IF	ID	EX	MEM									
LW						IF	ID	EX	MEM	WB							
ADD							IF	ID	EX	MEM	WB						
ADD								IF	ID	EX	MEM	WB					
SW									IF	ID	EX	MEM	WB				
BNEZ										IF	ID	EX	MEM	WB			
LW													IF	ID	EX	MEM	WB

Πλέον, η καθεμία από τις πρώτες 191 επαναλήψεις διαρκεί 12 κύκλους και η τελευταία διαρκεί 14. Συνεπώς, ο συνολικός αριθμών κύκλων επεξεργαστή είναι ίσος με $191 \cdot 12 + 14 = 2306$.