ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Μάθημα: Αρχιτεκτονική Υπολογιστών

Ονοματεπώνυμο: Ειρήνη Δόντη

Α.Μ: 03119839

2η Σειρά Γραπτών Ασκήσεων

5ο Εξάμηνο

Τμήμα 1ο

Αθήνα

2021 - 2022

Άσκηση 1

a)

- Ο προσωπικός δεκαδικός αριθμός είναι: 19,839.

Καταρχάς, πρέπει να δημιουργήσουμε τη μορφή υποδιαστολής IEEE. Η ζητούμενη μορφή είναι η παρακάτω:

S (1 bit)	Εκθέτης (5 bit)	Κλάσμα (10 bit)
0	10011	0011 1101 10

Για να υπολογίσουμε την παραπάνω μορφή, εκτελέσαμε τα παρακάτω βήματα:

Βήμα 1ο: Το S bit προσήμου είναι 0, γιατί αναφερόμαστε σε θετικό αριθμό.

Βήμα 2ο: Ο δεκαδικός αναπαρίσταται με τον εξής δυαδικό αριθμό:

1001 1.1101 0110 1100 1000 1011 0100 0011 1001 0101 1000 0001 0000 0110 0010 0100 1101

Βήμα 3ο: Κάνουμε κανονικοποίηση για να καθορίσουμε το κλάσμα:

1.0011 1101 0110 1100 1000 1011 0100 0011 1001 0101 1000 0001 0000 0110 0010 0100 1101 x 2^4

Βήμα 4ο: Καθορίζουμε την πόλωση του εκθέτη:

Η πόλωση είναι ίση με $2^{(5-1)} - 1 = 15$

Οπότε ο εκθέτης είναι ίσος με: 4+ πόλωση = $(19)_{10} = (10011)_2$

Βήμα 5ο: Αφαιρούμε το μοναδιαίο bit που είναι πιο αριστερά:

0011 1101 0110 1100 1000 1011 0100 0011 1001 0101 1000 0001 0000 0110 0010 0100 1101 x 2^4

Βήμα 6ο:Στρογγυλοποιούμε το παραπάνω αποτέλεσμα στα 10 bit:

Επειδή το 10ο bit στο παραπάνω αποτέλεσμα είναι μονάδα, για να στρογγυλοποιήσουμε τον αριθμό, θα προσθέσουμε το δυαδικό $(1)_2$.

Οπότε, το κλάσμα είναι: 0011 1101 10

- Ο μέγιστος αριθμός ο οποίος μπορεί να παρασταθεί στο δοσμένο 16-bit πρότυπο ΑΚΥ είναι:

Ο εκθέτης e παίρνει τιμές: $0 < e < 2^5 - 1 = 31$. Οπότε:

Max num:

$0\ 11110\ 1111111111 = (2^{emax - 15})(1{,}1111111111) = (2^{15})(2 - 2^{-10}) = 65504 \approx 2^{16}$

- Αντίθετα, ο απολύτως ελάχιστος αριθμός ο οποίος μπορεί να παρασταθεί στο ως άνω 16-bit πρότυπο ΑΚΥ είναι:

Min num:

$0\ 00001\ 0000000000 = (2^{1 - 15})(1{,}0000000000) = 2^{-14}$

MIN num:

$0\ 00000\ 0000000001 = (2^{1 - 15})(2^{-10}) = 2^{-24}$

b)

Θεωρώντας το πρότυπο IEEE 754-2008 απλής ακρίβειας (32 bit), συμπληρώνουμε τον κάτωθι πίνακα στο εν λόγω πρότυπο:

Αριθμός(10)	**Πρόσημο(2)**	**Εκθέτης(2)**	**Κλάσμα(2)**
0,00018	0	01110010	01111001011111001100010
-82	1	10000101	01001000000000000000000
0,0123	0	01111000	10010011000010111110000

Για να βρούμε τις απαντήσεις, ακολουθήσαμε τα εξής βήματα:

1ος Αριθμός [0,00018]:

Βήμα 1ο: Το S bit προσήμου είναι 0, γιατί αναφερόμαστε σε θετικό αριθμό.

Βήμα 2ο: Ο δεκαδικός αναπαρίσταται με τον εξής δυαδικό αριθμό:

0.0000 0000 0000 1011 1100 1011 1110 0110 0001 1100 1111 1111 1110 1011 0000 0111

Βήμα 3ο: Κάνουμε κανονικοποίηση για να καθορίσουμε το κλάσμα

$1.011\ 1100\ 1011\ 1110\ 0110\ 0001\ 1100\ 1111\ 1111\ 1110\ 1011\ 0000\ 0111 \times 2^{-13}$

Βήμα 4ο: Καθορίζουμε την πόλωση του εκθέτη:

Εφόσον αναφερόμαστε σε πρότυπο απλής ακρίβειας, τότε η πόλωση του εκθέτη είναι: 127

Συνεπώς, ο εκθέτης είναι -13 + πόλωση = $(114)_{10} = (01110010)_2$.

Βήμα 5ο: Αφαιρούμε το μοναδιαίο bit που είναι πιο αριστερά:

$011\ 1100\ 1011\ 1110\ 0110\ 0001\ 1100\ 1111\ 1111\ 1110\ 1011\ 0000\ 0111 \times 2^{-13}$

Βήμα 6ο: Στρογγυλοποιούμε το παραπάνω αποτέλεσμα στα 23 bit.

Επειδή το 23ο bit στο παραπάνω αποτέλεσμα είναι μονάδα, τότε για να στρογγυλοποιήσουμε τον αριθμό, θα προσθέσουμε το δυαδικό $(1)_2$.

Οπότε, το κλάσμα είναι: 011 1100 1011 1110 0110 0010

2ος Αριθμός [-82]:

Βήμα 1ο: Το S bit προσήμου είναι 1, γιατί αναφερόμαστε σε αρνητικό αριθμό.

Βήμα 2ο: Ο θετικός αριθμός αναπαρίσταται με τον εξής δυαδικό αριθμό:

1010010

Βήμα 3ο: Κάνουμε κανονικοποίηση για να καθορίσουμε το κλάσμα

1.010010×2^6

Βήμα 4ο: Καθορίζουμε την πόλωση του εκθέτη:

Εφόσον αναφερόμαστε σε πρότυπο απλής ακρίβειας, τότε η πόλωση του εκθέτη είναι: 127
Συνεπώς, ο εκθέτης είναι 6 + πόλωση = $(133)_{10} = (10000101)_2$.

Βήμα 5ο: Αφαιρούμε το μοναδιαίο bit που είναι πιο αριστερά:

010010×2^6.

Οπότε, το κλάσμα που προκύπτει είναι 01001000000000000000000

3ος Αριθμός [0,0123]:

Βήμα 1ο:Το S bit προσήμου είναι 0, γιατί αναφερόμαστε σε θετικό αριθμό.

Βήμα 2ο:Ο δεκαδικός αναπαρίσταται με τον εξής δυαδικό αριθμό:

0.0000 0011 0010 0110 0001 0111 1100 0001 1011 1101 1010 0101 0001 0001 1001 1100

Βήμα 3ο:Κάνουμε κανονικοποίηση για να καθορίσουμε το κλάσμα

$1.1\ 0010\ 0110\ 0001\ 0111\ 1100\ 0001\ 1011\ 1101\ 1010\ 0101\ 0001\ 0001\ 1001\ 1100 \times 2^{-7}$

Βήμα 4ο: Καθορίζουμε την πόλωση του εκθέτη:

Εφόσον αναφερόμαστε σε πρότυπο απλής ακρίβειας, τότε η πόλωση του εκθέτη είναι: 127

Συνεπώς, ο εκθέτης είναι -7 + πόλωση = $(120)_{10}$ = $(01111000)_2$.

Βήμα 5ο: Αφαιρούμε το μοναδιαίο bit που είναι πιο αριστερά:

1 0010 0110 0001 0111 1100 0001 1011 1101 1010 0101 0001 0001 1001 1100 x 2^{-7}

Βήμα 6ο:Στρογγυλοποιούμε το παραπάνω αποτέλεσμα στα 23 bit.

Επειδή το 23ο bit στο παραπάνω αποτέλεσμα είναι μονάδα, τότε για να στρογγυλοποιήσουμε τον αριθμό, θα προσθέσουμε το δυαδικό $(1)_2$.

Οπότε, το κλάσμα είναι:

1 0010 0110 0001 0111 1100 00

Άσκηση 2

Α)

α) Εξαρτήσεις δεδομένων:

I1: ADD **R1**,R2,R1 # ο καταχωρητής R1 γράφεται από την ADD.

I2: LW **R2**,0(**R1**) # η βάση R1 εξαρτάται από την εντολή I1.

I3: LW R1,4(**R1**) # η βάση R1 εξαρτάται από την εντολή I1.

I4: OR R3,R1,**R2** # ο πρώτος τελεστέος R1 εξαρτάται από την προηγούμενη εντολή # I3 και ο δεύτερος τελεστέος R2 εξαρτάται από την εντολή I2.

Η εντολή I2 εξαρτάται από την εντολή I1, καθώς η βάση R1 εξαρτάται από το αποτέλεσμα της εντολής I1. Επίσης, η εντολή I3 εξαρτάται από την εντολή I1, διότι η

βάση R1 εξαρτάται από το αποτέλεσμα της εντολής I1. Η εντολή I4 εξαρτάται από την εντολή I3 και την εντολή I2, διότι το αποτέλεσμα της εντολής I4 εξαρτάται από τα αποτελέσματα των εντολών I2 και I3.

b)

Δημιουργούμε το διάγραμμα χρονισμού το οποίο έχει στις στήλες τις εντολές και στις γραμμές τον χρόνο (σε κύκλους ρολογιού):

Χωρίς Προώθηση ο πίνακας συμπληρώνεται ως εξής:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12
I1	IF	ID	EX	MEM	WB							
I2		IF	ID	*	*	EX	MEM	WB				
I3			IF	*	*	ID	EX	MEM	WB			
I4				*	*	IF	ID	*	*	EX	MEM	WB

Παρατηρούμε ότι οι εντολές εκτελούνται μέσα σε 12 κύκλους ρολογιού.

Κίνδυνοι χωρίς προώθηση (περιμένουμε να ολοκληρωθεί η εντολή για να αντιμετωπίσουμε τον κίνδυνο δεδομένων):

Η εντολή I1 γράφει το αποτέλεσμα της στο 5^{ο} στάδιο, ενώ η εντολή I2 χρειάζεται να διαβάσει το αποτέλεσμα της εντολής I1 στο 3^{ο} στάδιο (read after write hazard). Γι' αυτό τον λόγο, για να αποφευχθούν οι κίνδυνοι δεδομένων, πρέπει να καθυστερήσουμε τον αλγόριθμο δύο κύκλους ρολογιού. Επίσης, η εντολή I3 χρειάζεται το αποτέλεσμα της εντολής I1 . Όμως, δε υπάρχει καθυστέρηση, γιατί ανάμεσα σε αυτές τις δύο εντολές παρεμβάλλονται πάνω από δύο κύκλοι ρολογιού και συνεπώς το αποτέλεσμα της εντολής I1 θα έχει καταχωρηθεί. Επιπλέον, τα αποτελέσματα των εντολών I3 και I2 χρησιμοποιούνται για τον υπολογισμό της εντολής I4. Η εντολή I3 γράφει το αποτέλεσμα στο 5^{ο} στάδιο, ενώ η εντολή I4 πρέπει

να διαβάσει το αποτέλεσμα αυτό στο 3ο στάδιο (read after write hazard). Γι' αυτό τον λόγο, για να αποφευχθούν οι κίνδυνοι δεδομένων, πρέπει να καθυστερήσει το πρόγραμμα δύο κύκλους ρολογιού. Η εντολή I2 έχει ήδη καταχωρήσει το αποτέλεσμα την στιγμή που η εντολή I4 το χρειάζεται, καθώς παρεμβάλλονται πάνω από δύο κύκλοι ρολογιού ανάμεσα στις εντολές αυτές. Οπότε, δε χρειάζεται να υπάρξει καθυστέρηση.

Με Προώθηση ο πίνακας συμπληρώνεται ως εξής:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
I1	IF	ID	EX	MEM	WB				
I2		IF	ID	EX	MEM	WB			
I3			IF	ID	EX	MEM	WB		
I4				IF	ID	*	EX	MEM	WB

Παρατηρούμε ότι οι εντολές εκτελούνται μέσα σε 9 κύκλους ρολογιού.

Κίνδυνοι με προώθηση (η κατάλληλη τιμή είναι διαθέσιμη πριν γραφεί στο αρχείο καταχωρητών):

Η προώθηση εμφανίζει κίνδυνο στις περιπτώσεις που μία εντολή προσπαθεί να διαβάσει έναν καταχωρητή ακριβώς μετά από μία εντολή lw που γράφει στον ίδιο καταχωρητή. Στην περίπτωση αυτή, η εντολή I4 χρησιμοποιεί το αποτέλεσμα της προηγούμενης εντολής I3 και συνεπώς, όπως διακρίνεται στον παραπάνω πίνακα, εμφανίζεται κίνδυνος δεδομένων και συνεπώς η εκτέλεση του αλγορίθμου θα επιβραδυνθεί κατά έναν κύκλο ρολογιού. Αυτό συμβαίνει επειδή τα αποτελέσματα

της εντολής lw είναι διαθέσιμα μετά το 4ο στάδιο , ενώ η εντολή (μορφής R) or χρειάζεται το αποτέλεσμα αυτό στο 3ο στάδιο (read after write hazard). Συνεπώς πρέπει να υπάρξει καθυστέρηση ώστε να αντιμετωπιστεί ο επονομαζόμενος κίνδυνος δεδομένων φόρτισης-χρήσης .

B)

a) Εξαρτήσεις δεδομένων:

I1: LW **R1**,0(R1) # ο καταχωρητής R1 ορίζεται από την LW

I2: AND **R1**,**R1**,R2 # ο πρώτος τελεστέος R1 εξαρτάται από την LW (εντολή I1)

I3: LW R2,0(**R1**) # η βάση R1 εξαρτάται από την εντολή AND (εντολή I2)

I4: LW R1,0(R3) # δεν υπάρχει κάποια εξάρτηση από τις προηγούμενες εντολές

Η εντολή I2 εξαρτάται από την εντολή I1, καθώς η τιμή του καταχωρητή R1 εξαρτάται από το αποτέλεσμα της εντολής I1. Επίσης, η εντολή I3 εξαρτάται από την εντολή I2, διότι η βάση R1 εξαρτάται από το αποτέλεσμα της εντολής I2. Η εντολή I4 δεν εξαρτάται από κάποια άλλη εντολή που εκτελείται νωρίτερα, γιατί το αποτέλεσμά της δεν εξαρτάται από το αποτέλεσμα κάποιας άλλης εντολής.

b)

Δημιουργούμε το διάγραμμα χρονισμού το οποίο έχει στις στήλες τις εντολές και στις γραμμές τον χρόνο (σε κύκλους ρολογιού):

Χωρίς Προώθηση ο πίνακας συμπληρώνεται ως εξής:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12
I1	IF	ID	EX	MEM	WB							
I2		IF	ID	*	*	EX	MEM	WB				
I3			IF	*	*	ID	*	*	EX	MEM	WB	
I4				*	*	IF	*	*	ID	EX	MEM	WB

Παρατηρούμε ότι οι εντολές εκτελούνται μέσα σε 12 κύκλους ρολογιού.

Κίνδυνοι χωρίς προώθηση (περιμένουμε να ολοκληρωθεί η εντολή για να αντιμετωπίσουμε τον κίνδυνο δεδομένων):

Η εντολή I1 γράφει το αποτέλεσμα της στο 5^{ο} στάδιο, ενώ η εντολή I2 χρειάζεται το αποτέλεσμα της εντολής I1 στο 3^{ο} στάδιο (read after write hazard). Γι' αυτό τον λόγο, για να αποφευχθούν οι κίνδυνοι δεδομένων, πρέπει να καθυστερήσει το πρόγραμμα δύο κύκλους ρολογιού. Επίσης, η εντολή I3 χρειάζεται στο 3^{ο} στάδιο το αποτέλεσμα της εντολής I2, το οποίο καταχωρείται στο 5^{ο} στάδιο (read after write hazard). Οπότε, όμοια με πριν, πρέπει να υπάρξει καθυστέρηση δύο κύκλων για να αποφευχθούν οι κίνδυνοι δεδομένων. Η εντολή I4 δεν εξαρτάται από κάποια εντολή, καθώς η βάση R3 δεν τίθεται υπό επεξεργασία στις αμέσως προηγούμενες εντολές.

Με Προώθηση ο πίνακας συμπληρώνεται ως εξής:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9
I1	IF	ID	EX	MEM	WB				
I2		IF	ID	*	EX	MEM	WB		
I3			IF	*	ID	EX	MEM	WB	
I4				*	IF	ID	EX	MEM	WB

Παρατηρούμε ότι οι εντολές εκτελούνται σε 9 κύκλους ρολογιού.

Κίνδυνοι με προώθηση (η κατάλληλη τιμή είναι διαθέσιμη πριν γραφεί στο αρχείο καταχωρητών):

Η προώθηση εμφανίζει κίνδυνο στις περιπτώσεις που μία εντολή προσπαθεί να διαβάσει έναν καταχωρητή ακριβώς μετά από μία εντολή lw που γράφει στον ίδιο καταχωρητή. Στην περίπτωση αυτή, η εντολή I2 χρησιμοποιεί το αποτέλεσμα της προηγούμενης εντολής I1 και συνεπώς, όπως διακρίνεται στον παραπάνω πίνακα, εμφανίζεται κίνδυνος δεδομένων ο οποίος θα καθυστερήσει την εκτέλεση κατά έναν κύκλο ρολογιού. Αυτό συμβαίνει επειδή τα αποτελέσματα της εντολής lw είναι διαθέσιμα μετά το 4^{ο} στάδιο , ενώ η εντολή (μορφής R) and χρειάζεται το αποτέλεσμα αυτό στο 3^{ο} στάδιο (read after write hazard). Συνεπώς πρέπει να υπάρξει καθυστέρηση ώστε να αντιμετωπιστεί ο επονομαζόμενος κίνδυνος δεδομένων φόρτισης-χρήσης .

Άσκηση 3

Η αρχική τιμή του καταχωρητή $s3 είναι ο δεκαεξαδικός $(0x139)_{16} = (313)_{10}$

```
I1:      add $t5, $zero, $zero
I2:      add $t0, $zero, $zero
I3: L:   sll $t1, $t0, 2
I4:      add $t2, $t1, $s1
I5:      lw $t4, 0($t2)
I6:      add $t3, $t1, $s2
I7:      add $t5, $t5, $t3
I8:      add $t4, $t4, $t5
I9:      sw $t4, 0($t2)
I10:     addi $t0, $t0, 1
I11:     slt $t6, $t0, $s3
I12:     bne $t6, $zero, L
```

(1)

Θεωρούμε ότι δεν υπάρχουν σχήματα προώθησης, οπότε δημιουργούμε το παρακάτω διάγραμμα χρονισμού για την $1^{η}$ επανάληψη του βρόχου:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15
I1	IF	ID	EX	MEM	WB										
I2		IF	ID	EX	MEM	WB									
I3			IF	ID	*	*	EX	MEM	WB						
I4				IF	*	*	ID	*	*	EX	MEM	WB			
I5					*	*	IF	*	*	ID	*	*	EX	MEM	WB
I6					*	*		*	*	IF	*	*	ID	EX	MEM
I7					*	*		*	*		*	*	IF	ID	*
I8					*	*		*	*		*	*		IF	*
I9					*	*		*	*		*	*			*
I10					*	*		*	*		*	*			*
I11					*	*		*	*		*	*			*
I12					*	*		*	*		*	*			*
I3					*	*		*	*		*	*			*

	CC16	CC17	CC18	CC19	CC20	CC21	CC22	CC23	CC24	CC25	CC26	CC27	CC28	CC29	CC30	CC31	CC32	CC33	CC34	CC35	CC36
I6	WB																				
I7	*	EX	MEM	WB																	
I8	*	ID	*	*	EX	MEM	WB														
I9	*	IF	*	*	ID	*	*	EX	MEM	WB											
I10	*		*	*	IF	*	*	ID	EX	MEM	WB										
I11	*		*	*		*	*	IF	ID	*	*	EX	MEM	WB							
I12	*		*	*		*	*		IF	*	*	*	*	ID	EX	MEM	WB				
I3	*		*	*		*	*			*	*	*	*				IF	ID	EX	MEM	WB

Λόγοι οποιαδήποτε παρατηρούμενης καθυστέρησης:

Η εντολή I2 γράφει το αποτέλεσμα της στο 5ο στάδιο, ενώ η εντολή I3 χρειάζεται το αποτέλεσμα της εντολής I2 στο 3ο στάδιο. Γι' αυτό τον λόγο, για να αποφευχθούν οι κίνδυνοι δεδομένων, πρέπει το πρόγραμμα να καθυστερήσει κατά δύο κύκλους ρολογιού ανάμεσα στα στάδια ID και EX της εντολής I3 . Επίσης, η εντολή I4 χρειάζεται στο 3ο στάδιο το αποτέλεσμα της εντολής I3, το οποίο καταχωρείται στο 5ο στάδιο. Οπότε, όμοια με πριν, πρέπει να υπάρξει καθυστέρηση δύο κύκλων ρολογιού ανάμεσα στα στάδια ID και EX της εντολής I4 για να αποφευχθούν οι κίνδυνοι δεδομένων. Όμοια με πριν, πρέπει να υπάρξει η καθυστέρηση δύο κύκλων ρολογιού ανάμεσα στα στάδια ID και EX εξαιτίας των εξαρτήσεων ανάμεσα στα ζεύγη εντολών: I4 – I5, I6 - I7, I7 - I8, I8 - I9 και I10 – I11.

Στην εντολή I12 ο αλγόριθμος αποφαίνεται αν θα συνεχίσει τον αλγόριθμο σειριακά ή θα επιστρέψει στη διακλάδωση L. Ο επεξεργαστής, σε αυτήν την περίπτωση, πρέπει να εισαγάγει (στις περιπτώσεις που εισάγεται στη διακλάδωση L) καθυστερήσεις μέχρι την επίλυση η οποία πραγματοποιείται στο στάδιο EX.

Ο βρόχος L εκτελείται 313 φορές (η συνθήκη της επανάληψης εκτελείται για i < 313, ενώ εντός κάθε βρόχου i +=1 στο παραπάνω πρόγραμμα), οπότε απαιτούνται:

312*31 + 1*32 = 9704 κύκλοι για την εκτέλεση ολόκληρου του κώδικα.

(2)

Θεωρούμε ότι υπάρχουν σχήματα προώθησης, οπότε δημιουργούμε το παρακάτω διάγραμμα χρονισμού για την 1η επανάληψη του βρόχου:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20
I1	IF	ID	EX	MEM	WB															
I2		IF	ID	EX	MEM	WB														
I3			IF	ID	EX	MEM	WB													
I4				IF	ID	EX	MEM	WB												
I5					IF	ID	EX	MEM	WB											
I6						IF	ID	EX	MEM	WB										
I7							IF	ID	EX	MEM	WB									
I8								IF	ID	EX	MEM									
I9									IF	ID	EX	MEM	WB							
I10										IF	ID	EX	MEM	WB						
I11											IF	ID	EX	MEM	WB					
I12												IF	ID	EX	MEM	WB				
I3																IF	ID	EX	MEM	WB

Υπόδειξη προωθήσεων:

Η προώθηση εμφανίζει κίνδυνο στις περιπτώσεις που μία εντολή προσπαθεί να διαβάσει έναν καταχωρητή ακριβώς μετά από μία εντολή lw που γράφει στον ίδιο καταχωρητή. Η εντολή I3 πρέπει να χρησιμοποιήσει το αποτέλεσμα της προηγούμενης εντολής I2 στο 3ο στάδιο, ενώ η εντολή I2 υπολογίζει το αποτέλεσμά της μετά το 3ο στάδιο. Οπότε, με το σχήμα της προώθησης, δεν απαιτείται κάποιος κύκλος καθυστέρησης για την μετάβαση από την εντολή I2 στην εντολή I3, αφού δεν χρειάζεται να αποθηκευτεί το αποτέλεσμα στον καταχωρητή για να υπολογιστεί το αποτέλεσμα της επόμενης εντολής που χρησιμοποιεί την τιμή του καταχωρητή . Το ίδιο συμβαίνει και με τα ζεύγη εντολών I3-I4, I4-I5, I6-I7, I7-I8, I10-I11. Για το

ζεύγος εντολών I8 – I9, ισχύει ότι η εντολή I9 (sw) χρειάζεται το αποτέλεσμα της εντολής I9 (του καταχωρητή $t4) στο 4^{ο} στάδιο. Από την άλλη, η εντολή I8 υπολογίζει το αποτέλεσμα των πράξεων της στο 3^{ο} στάδιο, και συνεπώς, αφού υπάρχουν όλα τα σχήματα προώθησης, δεν χρειάζεται κάποια καθυστέρηση κύκλου ρολογιού.

Ο βρόχος L εκτελείται 313 φορές (η συνθήκη της επανάληψης εκτελείται για i < 313, ενώ εντός κάθε βρόχου i +=1 στο παραπάνω πρόγραμμα), οπότε απαιτούνται:

312*15 + 1*16 = 4696 κύκλοι για την εκτέλεση ολόκληρου του κώδικα.

(3)

Θεωρούμε την ίδια σωλήνωση με το ερώτημα (2). Δεν χρειάζεται να αναδιατάξουμε τον κώδικα (χωρίς να αλλάξουμε τη σημασιολογία του προγράμματος), γιατί, όπως προαναφέραμε, στο ερώτημα (2) δεν υπάρχουν ζεύγη εντολών που να εμφανίζουν κίνδυνο και συνεπώς δεν υπάρχει κάποια καθυστέρηση στο πρόγραμμα σε κύκλους ρολογιού. Οπότε, δε χρειάζεται να αναδιατάξουμε τον κώδικα για να αποφύγουμε κάποια καθυστέρηση στο πρόγραμμα.

Το διάγραμμα χρονισμού είναι το ίδιο με το προηγούμενο ερώτημα:

	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20
I1	IF	ID	EX	MEM	WB															
I2		IF	ID	EX	MEM	WB														
I3			IF	ID	EX	MEM	WB													
I4				IF	ID	EX	MEM	WB												
I5					IF	ID	EX	MEM	WB											
I6						IF	ID	EX	MEM	WB										
I7							IF	ID	EX	MEM	WB									
I8								IF	ID	EX	MEM									
I9									IF	ID	EX	MEM	WB							
I10										IF	ID	EX	MEM	WB						
I11											IF	ID	EX	MEM	WB					
I12												IF	ID	EX	MEM	WB				
I3																IF	ID	EX	MEM	WB

Όπως με το προηγούμενο ερώτημα, απαιτούνται 312*15 + 1*16 = 4696 κύκλοι για την εκτέλεση ολόκληρου του κώδικα.