NTUA DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Άσκηση 2η: Assembly MIPS

ΜΑΘΗΜΑ ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ

Μαρίνα Φραγκούλη

12/9/2024

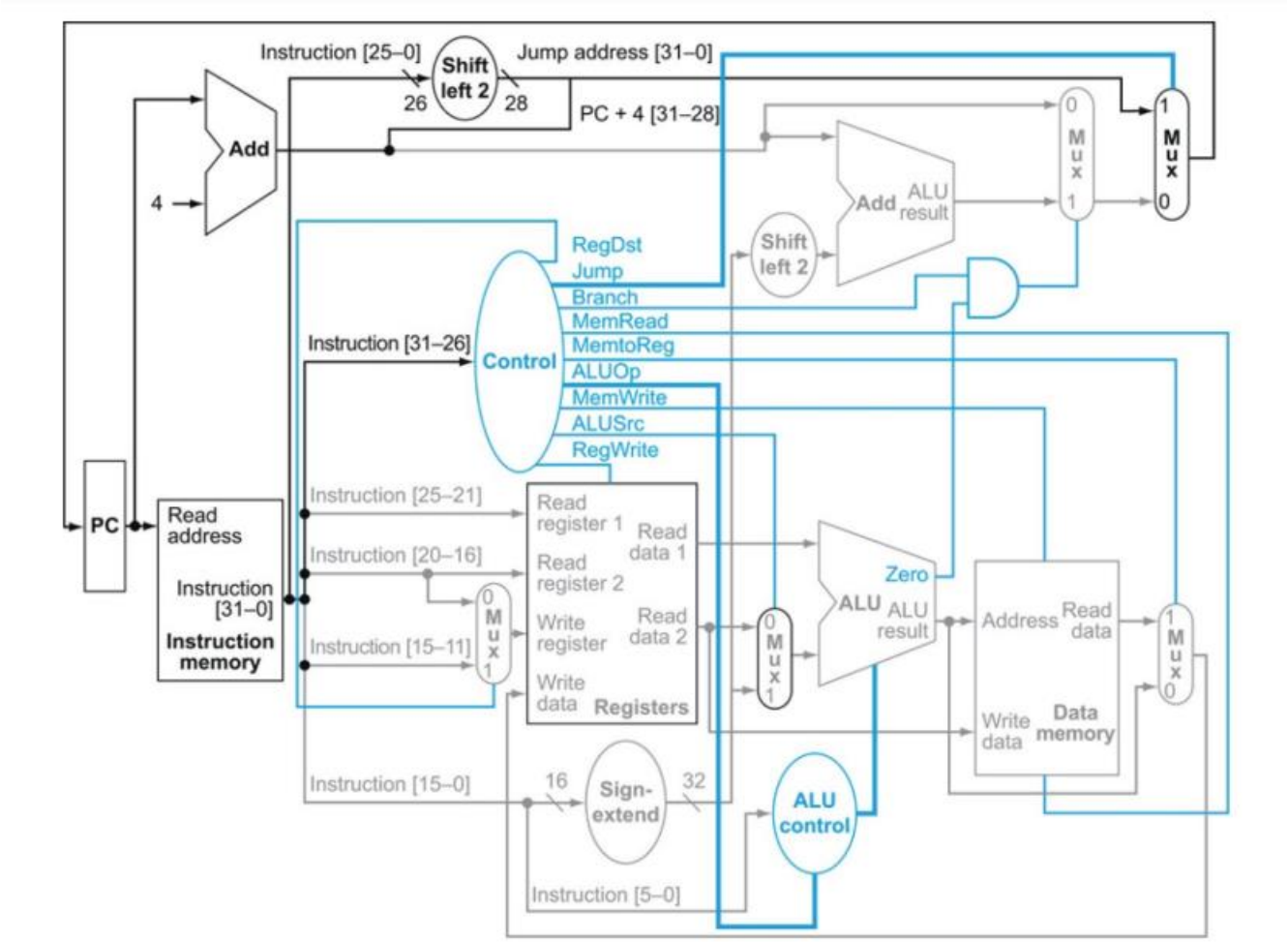
Τμήμα 3 (ΠΑΞ-Ω)

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΜΕΡΟΣ Α

1) ΠΕΡΙΓΡΑΦΗ ΤΟΥ DATAPATH ΤΗΣ ΕΚΦΩΝΗΣΗΣ ΓΙΑ BNE

Από την εκφώνηση:



Για το δοσμένο Datapath, ας δούμε την ροή δεδομένων για την εντολή bne σύμφωνα με τον single cycle κύκλο εντολών 5 σταδίων:

IF | ID | EX | MEM | WB

Instruction Fetch (IF)

Ο PC διαβάζει την registered εντολή και υπολογίζει την διεύθυνση της επόμενης με PC+4. Στο τέλος του IF στην έξοδο του ADDER 1 έχουμε την διεύθυνση του PC +4

Instruction Decode (ID)

Ο σκοπός αυτού του βήματος του κύκλου εντολών είναι να αποκωδικοποιήσει την εντολή και να διαβάσει (στην περίπτωση του bne αφού έχει δύο ορίσματα) δύο καταχωρητές-ορίσματα τους $rs και $rt.

Το Instruction Memory Unit διαβάζει από την έξοδο του PC την διεύθυνση της τρέχουσας εντολής. Στην έξοδό του μας δίνει το περιεχόμενο της διεύθυνσης που δείχνει το PC δηλαδή το binary της τρέχουσας εντολής (Instruction[31:0]). Διαφορετικά τμήματα του Instruction[31:0] ακολουθούν διαφορετική διαδρομή.

Αυτά τα πρώτα δύο βήματα είναι ίδια για κάθε εντολή

Execution (EX)

Αυτό το στάδιο είναι το κύριο στάδιο διαφοροποίησης από εντολή σε εντολή. Σε αυτό το στάδιο δημιουργείται το πρόβλημα όπως θα δούμε παρακάτω.

Memory Access (MEM)

Το στάδιο αυτό αφορά μόνο εντολές που εκτελούν πρόσβαση στην μνήμη δηλαδή lw, sw. Το αγνοούμε αφού ασχολούμαστε μόνο με την εκτέλεση της bne.

Write Back (WB)

Το στάδιο αυτό αφορά μόνο εντολές που εκτελούν ενημερώνουν τον καταχωρητή προορισμού δηλαδή lw. Της το αγνοούμε αφού ασχολούμαστε μόνο με την εκτέλεση της bne.

Instruction Fetch (IF)'

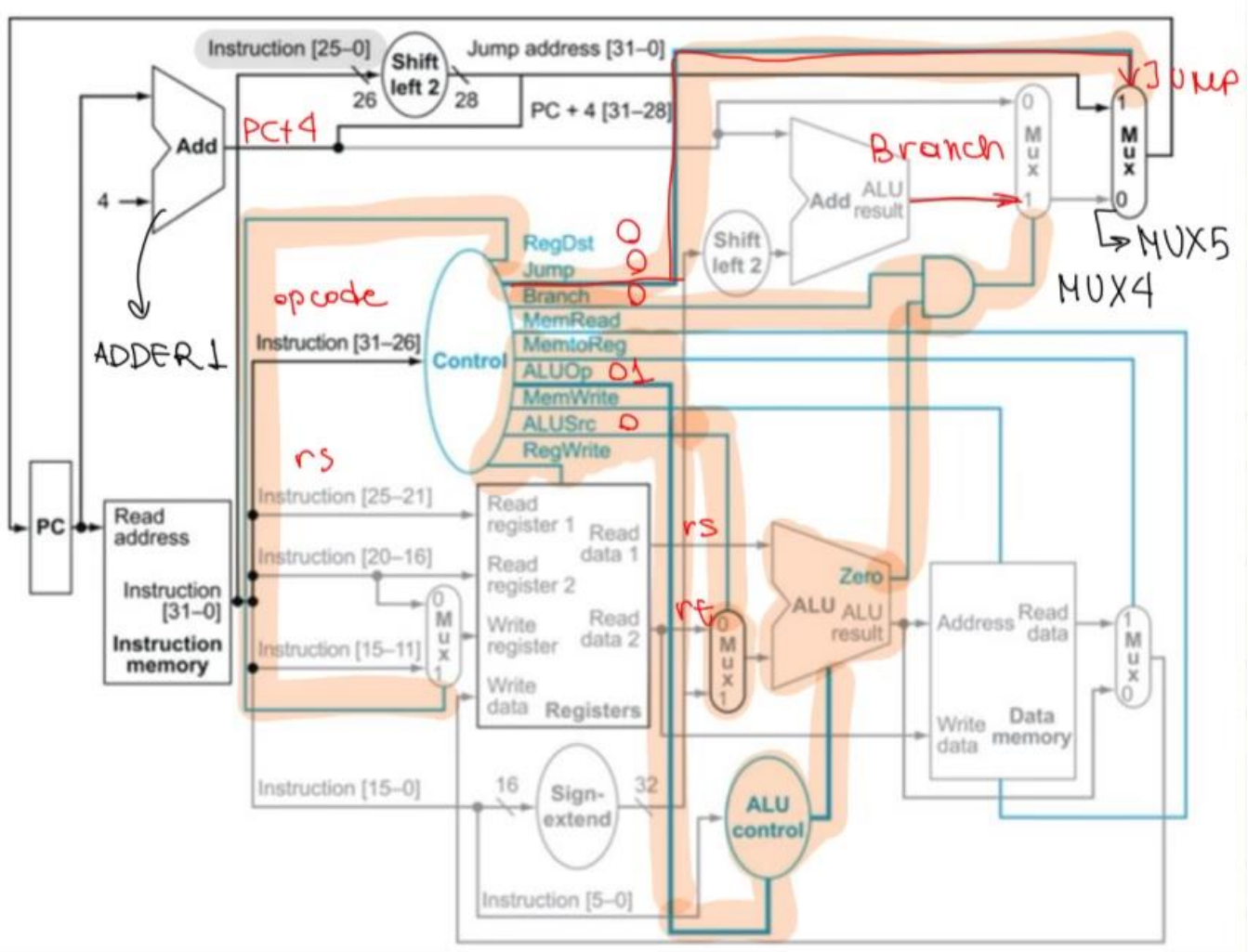
Ο κύκλος ξεκινάει από την αρχή.

.

.

.

Όπως αναφέραμε θα λοιπόν θα επικεντρωθούμε στο EX στάδιο:



Ήδη από το ID έχουμε το Instruction[31:0] τα κατάλληλα bits του οποίου «πηγαίνουμε» στα κατάλληλα σημεία του συνδυαστικού μας κυκλώματος. Συγκεκριμένα:

Instruction[31:26] (opcode)

περιέχει το opcode της εντολής δηλαδή το αναγνωριστικό της, για κάθε τύπο εντολής (I, R, J) και πηγαίνει στην είσοδο του συνδυαστικού λογικού κυκλώματος Control Unit. Αυτό με την σειρά του παράγει τα κατάλληλα output για κάθε μια από τις εξόδους του.

Οι παράμετροι από την έξοδο του Control που μας ενδιαφέρουν στο bne είναι:

0:RegDst = 0 δηλαδή έχω I type εντολή Instruction[20:16] ως rt,

1: Jump = 0 δηλαδή δεν έχω Jump,

2: Branch = 0 δεν έχω beq έχω bne,

5-6: ALUOp δηλαδή 01 (αφαίρεση) για beq, bne (έτσι η ALU κάνει την σύγκριση των rs, rt) και

8: ALUSrc =0 αφού για bec έχω I-type εντολή και πρέπει να διαβάσω Read Data 2.

Instruction[25:21] (επειδή bne I-Type αντιστοιχεί στο rs register-όρισμα 1)

Περιέχει τον rs δηλαδή την διεύθυνση του register που περιέχει το όρισμα 1. Το register unit χρησιμοποιεί την διεύθυνση αυτή για να διαβάσει και να περάσει στην έξοδό του Read Data 1 το περιεχόμενου του rs.

Instruction[20:16] (επειδή bne I-Type αντιστοιχεί στο rt register-όρισμα 1)

Περιέχει τον rt δηλαδή την διεύθυνση του register που περιέχει το όρισμα 2. Το register unit χρησιμοποιεί την διεύθυνση αυτή για να διαβάσει και να περάσει στην έξοδό του Read Data 2 το περιεχόμενου του rt.

Τα Read Data 1 και Read Data 2 όταν είναι έτοιμα μπαίνουν στην ALU και περιμένουν μέχρι να χρησιμοποιηθούν για την σύγκριση.

Instruction [15:0] (επειδή bne αντιστοιχεί στο label)

Το label είναι συμβολικός δείκτης σε διεύθυνση μιας εντολής στον κώδικα. Ο compiler γνωρίζει την πραγματική διεύθυνση στην μνήμη που αντιστοιχεί στο label ωστόσο για χώρους εξοικονόμησης χώρου (δεδομένου ότι πρέπει να χωράει στο 16bits καλούπι των I-Type εντολών) ο compiler (ή πιο ακριβέστερα ο assembler) χρησιμοποιεί το label για να μας δώσει το offset (in words).

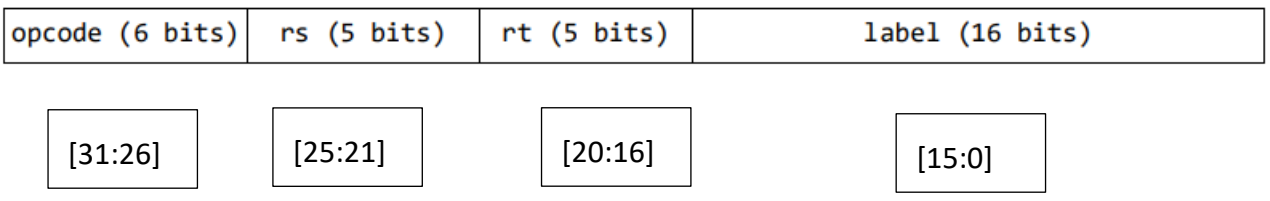
Όλα τα παραπάνω δουλεύουν χωρίς καμία αλλαγή επειδή το δοσμένο datapath έχει ενσωματωμένη την εντολή branch on equal (και μάλιστα εκτεταμένη για την εντολή άλματος jump).

Λόγω του ότι οι πολυπλέκτες 4 και 5 είναι συνδεδεμένοι η έξοδος το 4 στην μια είσοδο του 5 καταλαβαίνουμε ότι το datapath της branch σχετίζεται στενά με το datapath της jump.

2) ΥΠΟΕΡΩΤΗΜΑ 1

1) Εξηγήστε γιατί το datapath αυτό δεν υποστηρίζει την εκτέλεση της εντολής branch-on-not-equal. Σας υπενθυμίζουμε τη μορφή της εντολής bne (στις παρενθέσεις δίνεται το μήκος του κάθε πεδίου), η οποία έχει opcode = 0x5.

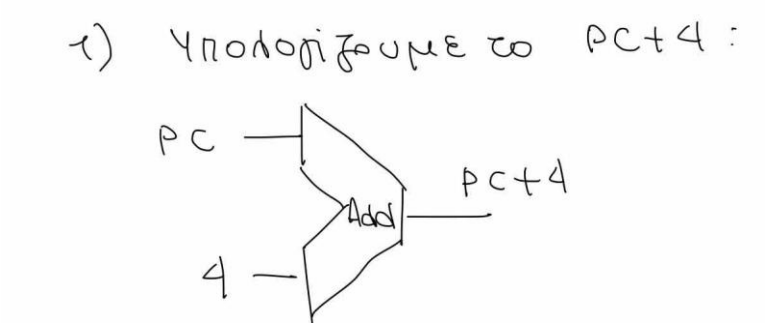
bne rs, rt, label



Που το πρόβλημα ?

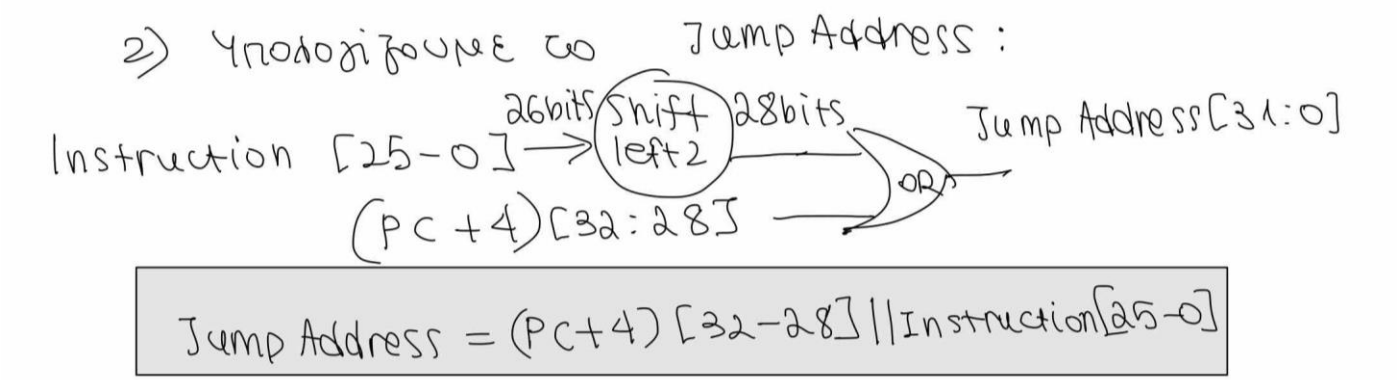
Το νόημα της εντολής bne (όπως και του beq και του j) είναι η σωστή ενημέρωση του PC στο επόμενο IF. Για τον σκοπό αυτό υπολογίζονται από τα κατάλληλα συνδυαστικά τα :

Αρχικά (PC+4):

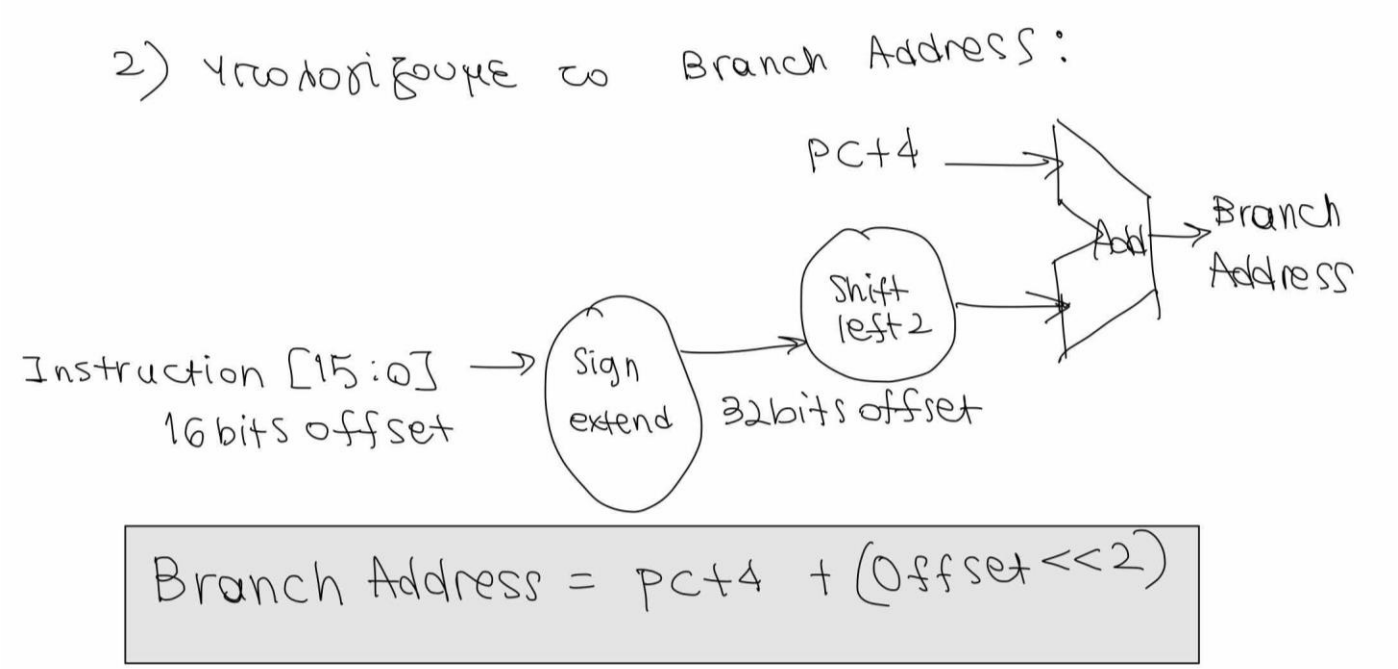


(Αυτό υπάρχει ήδη από το τέλος του IF ενώ όλα τα υπόλοιπα γίνονται στο EX.)

Και στην συνέχεια μετά το ID κατά το EX υπολογίζεται η Jump Address:

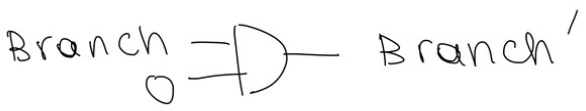


Και παράλληλα η Branch Address:

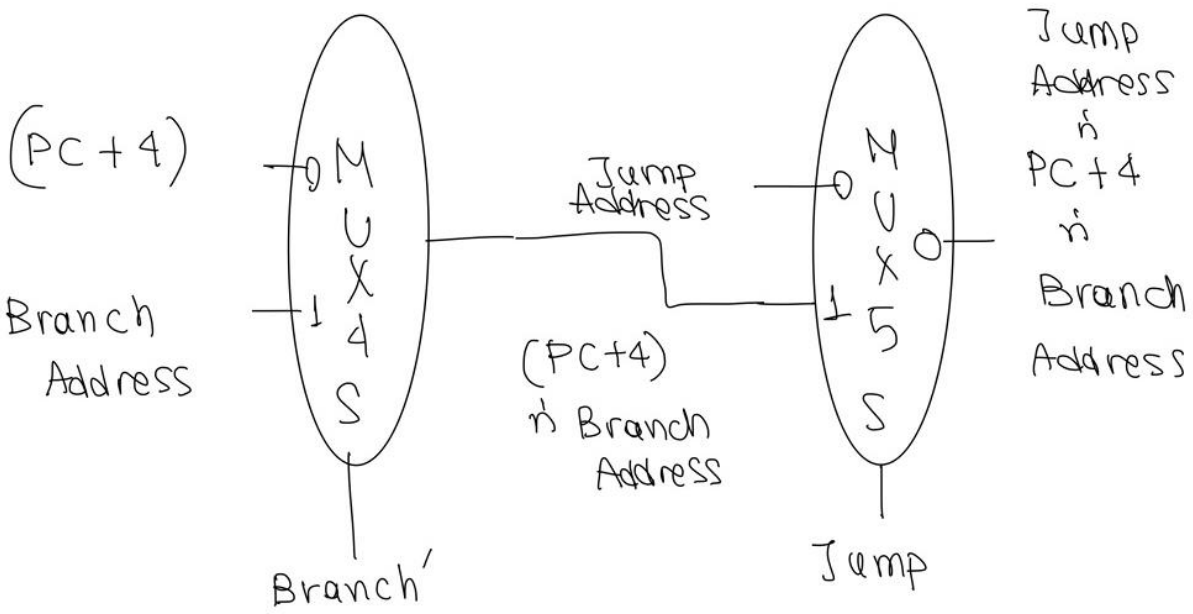


Το instruction [15:0] είναι 16 bit offset επειδή beq και ο sign extend κάνει το 16 bit offset 32 για να μπορώ να κάνω πράξεις

Επομένως τώρα πάμε στους πολυπλέκτες. Το Select Bit του Mux 4 είναι



Ενώ το select bit του MUX 5 είναι το Jump. Έτσι για την έξοδο του MUX 5 που θα γίνει είσοδος στο PC έχουμε το απλοποιημένο κύκλωμα:



Που δίνει τον ακόλουθο πίνακα

Jump	Branch	MUX 5 OUT	Σχόλια
0	0	PC+4	Όχι jump, όχι branch άρα PC→(PC+4)
0	1	Branch Address	Όχι jump, ναι branch άρα PC→(Branch Address)
1	0	Jump Address	Ναι jump, όχι branch άρα PC→(Jump Address)
1	1	-	Άτοπο

Από τον πίνακα αυτόν είναι εμφανές ότι η δοσμένη δομή δεδομένων δεν προβλέπει την εντολή beq αλλα είναι εφοδιασμένη με τις εντολές beq και jump. Αυτό οφείλεται σε δύο λόγους:

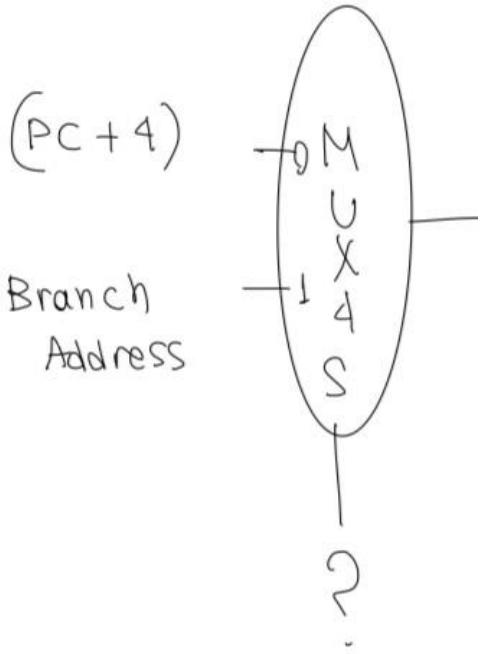
στο ότι ως Select bit στην είσοδο του MUX 4 χρησιμοποιείται το Branch AND Zero και δεν αξιοποιείται το αποτέλεσμα της σύγκρισης των rt, rs που υπολογίζεται στην έξοδο της ALU.

Στο ότι δεν υπάρχει κάπου έλεγχος για operant του beq δηλαδή δεν υπάρχει κάποιο bit ως έξοδο του Control Unit που να επιβεβαιώνει ότι πράγματι η εντολή που πραγματοποιείται είναι bne και όχι απλώς έλλειψη beq, jump.

3) ΥΠΟΕΡΩΤΗΜΑ 2

Από την εκφώνηση:

2) Περιγράψτε τις επεκτάσεις/τροποποιήσεις του datapath που απαιτούνται, ώστε να μπορεί να εκτελείται η bne.



Όπως είπαμε παραπάνω πρέπει στο Control Unit να προστεθεί κάποιο bit ως έξοδο του Control Unit που να επιβεβαιώνει ότι πράγματι η εντολή που πραγματοποιείται είναι bne και όχι απλώς έλλειψη beq, jump. Αυτό το bit θα το ονομάσουμε BranchNot.

Όσο αναφορά την σωστή επιλογή της νέας τιμής του PC, σύμφωνα με το ερώτημα 1, καταλαβαίνουμε ότι αρκεί να αλλάξουμε το Select bit του MUX 4. Οι δύο είσοδοι του MUX 4 παραμένουν ίδιες.

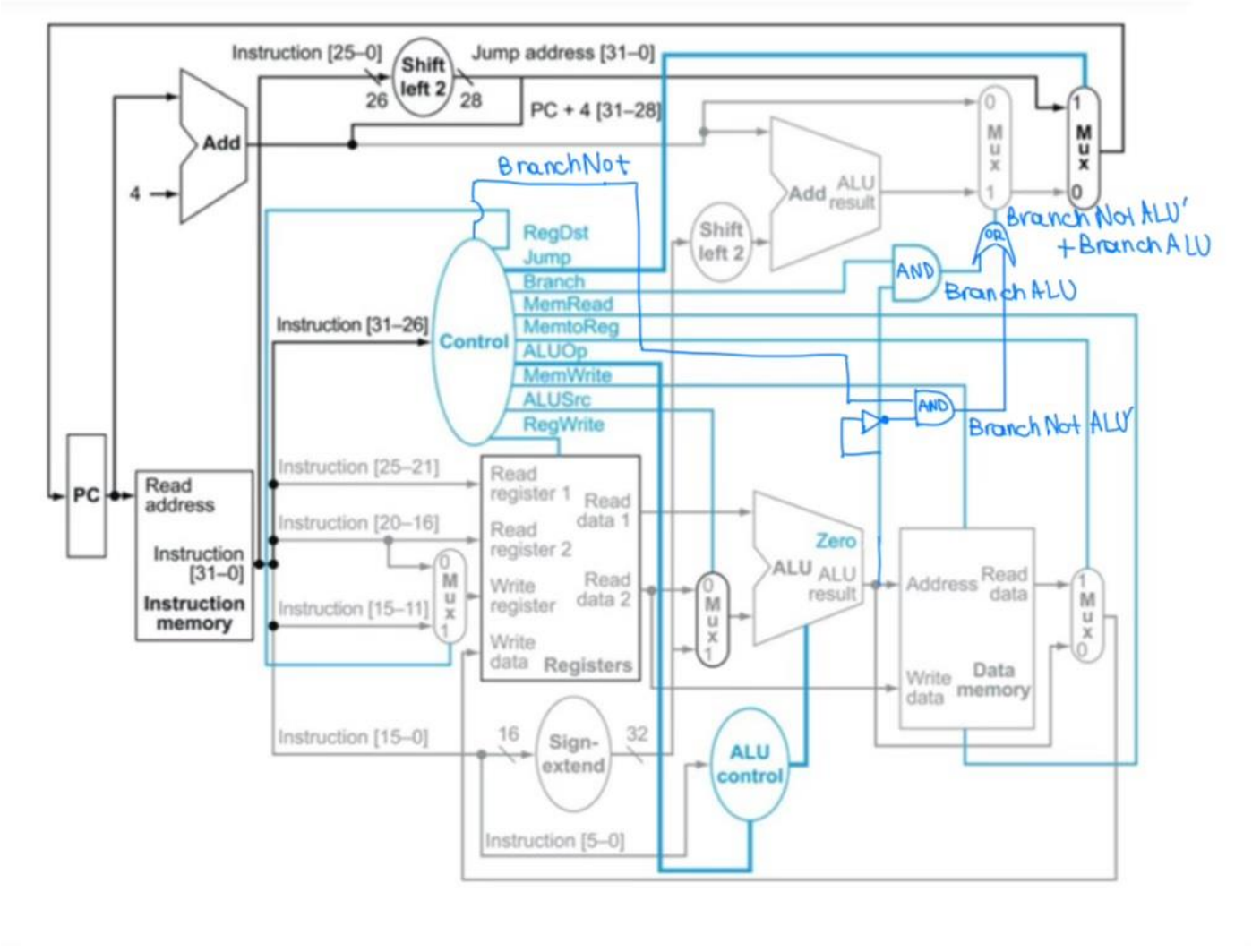
ALU result	Branch	BranchNot	Επιθυμητό Select Bit
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	x
1	1	1	x

Απλοποίηση συνδυαστικού κυκλώματος:

B\BN ALU	00	01	11	10
0	0	0	0	1
1	0	1	x	x

Επομένως: Select Bit MUX4=BranchNot ALU'+Branch ALU

Δηλαδή το συνδυαστικό κύκλωμα υλοποιείται με δύο AND, μία OR και μία NOT.



Η απλή μονάδα ελέγχου και η διαδρομή δεδομένων εκτεταμένες για τον χειρισμό της εντολής bne.

2. ΜΕΡΟΣ Β

1) ΕΚΦΩΝΗΣΗ ΚΑΙ ΠΑΡΑΔΟΧΕΣ

Δίνεται ο παρακάτω κώδικας :

```
1.   LOOP:   LW $t2, 8($t1)
2.            ADD $t2, $t2, $t1
3.            SW $t2, 8($t1)
4.            ADDI $t1, $t1, 8
5.            LW $t6, 8($t1)
6.            LW $t5, 16($t1)
7.            ADD $t2, $t6, $t5
8.            SW $t2, 8($t1)
9.            ADDI $t4, $t4, -4
10.           BNE $t9, $t4, LOOP
```

Δίνονται οι αρχικές τιμές των καταχωρητών $t9=0x1000 και $t4=0x1400. Υποθέστε την κλασική αρχιτεκτονική σωλήνωσης του MIPS αποτελούμενη από τα στάδια IF, ID, EX, MEM, WB. Όλα τα στάδια διαρκούν ένα κύκλο. Κατά τον εντοπισμό μιας εντολής άλματος υπό συνθήκη, ο επεξεργαστής κάνει stall τη σωλήνωση μέχρι την επίλυση, η οποία πραγματοποιείται στο στάδιο EX. Τέλος, υποθέστε ότι η εγγραφή σε ένα καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου.

2) ΥΠΟΕΡΩΤΗΜΑ 1 ΕΚΦΩΝΗΣΗ

1) Αρχικά, υποθέτουμε ότι η αρχιτεκτονική σωλήνωσης *δε διαθέτει σχήμα προώθησης* (forwarding). Για την 1η επανάληψη του παραπάνω βρόχου (μέχρι και την πρώτη εντολή της 2ης επανάληψης), συμπληρώστε ένα διάγραμμα χρονισμού όπως αυτό που παρουσιάζεται στη συνέχεια, για να δείξετε τα διάφορα στάδια του pipeline από τα οποία διέρχονται οι εντολές σε αυτό το διάστημα εκτέλεσης. Υποδείξτε και εξηγείστε τους πιθανούς κινδύνους (hazards) που μπορούν να προκύψουν κατά την εκτέλεση, καθώς και τον τρόπο με τον οποίον αυτοί αντιμετωπίζονται.

Κύκλος	*1*	*2*	*3*	*4*	*5*	*6*	...
Εντολή 1	IF	ID	EX	MEM	WB		
Εντολή 2		IF	ID	EX	MEM	WB	
Εντολή 3			...	...	...	...	...

Πόσοι κύκλοι απαιτούνται συνολικά για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η);

3) ΥΠΟΕΡΩΤΗΜΑ 1 ΕΞΗΓΗΣΗ HAZARDS ΚΑΙ ΤΡΟΠΟΙ ΑΝΤΙΜΕΤΩΠΙΣΗΣ

Γενικά υπάρχουν 3 είδη Hazards:

- Data Hazards
- Control Hazards
- Structural Hazards

Οι τρόποι αντιμετώπισης έγκειται στην μετατροπή ενός single cycle datapath σε pipeline. Εκτός από την εισαγωγή των ενδιάμεσων registers και τις κατάλληλες τροποποιήσεις στο ρολόι η μετατροπή αυτή χρειάζεται την προσθήκη ενός Hazard Control Unit. Αυτό το Unit θα έχει τρία subunits ένα για κάθε είδος κινδύνου. Στην περίπτωσή μας δηλαδή για μια αρχιτεκτονική RISC τύπου MIPS με υλοποίηση με pipeline:

- Για τα Data Hazards

  Οφείλονται σε μία από τις παρακάτω εξαρτήσεις δεδομένων:

  - RAW
  - WAR
  - WAW

  Λόγω της δομής του datapath MIPS (με τα 5 στάδια) μόνο εξαρτήσεις τύπου RAW μπορεί να δημιουργήσουν πρόβλημα και συγκεκριμένα μόνο οι παρακάτω 2 περιπτώσεις:
- Για τα Structural Hazard

  Το Structural Control Subunit μπορεί να εξαλειφθεί τελείως αν θέσουμε συχνότητα του ρολογιού κατάλληλη ώστε σε έναν κύκλο να μπορεί τόσο το Data Memory Unit όσο και το Register Unit να προλάβει στο μισό πρώτο του κύκλου να γράψει και στο δεύτερο μισό του κύκλου να διαβάσει. Στην εκφώνηση αναφέρει ότι *« ότι η εγγραφή σε ένα καταχωρητή γίνεται*

στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου.» επομένως δεν θα ασχοληθούμε με το Structural Control Subunit.

- Για τα Control Hazards
  Στην εκφώνηση αναφέρει ότι *«Κατά τον εντοπισμό μιας εντολής άλματος υπό συνθήκη, ο επεξεργαστής κάνει stall τη σωλήνωση μέχρι την επίλυση, η οποία πραγματοποιείται στο στάδιο EX.»* Αυτό επιλύει τα Control Hazards επομένως δεν θα ασχοληθούμε με το Control Hazards Control Subunit ωστόσο μία πιθανή υλοποίηση του είναι η εξής:

4) ΥΠΟΕΡΩΤΗΜΑ 1 ΔΙΑΓΡΑΜΜΑ ΧΡΟΝΙΣΜΟΥ

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM	WB												
Εντολή 2	ADD $t2, $t2, $t1		IF	ID	EX	MEM	WB											
Εντολή 3	SW $t2, 8($t1)			IF	ID	EX	MEM	WB										
Εντολή 4	ADDI $t1, $t1, 8				IF	ID	EX	MEM	WB									
Εντολή 5	LW $t6, 8($t1)					IF	ID	EX	MEM	WB								
Εντολή 6	LW $t5, 16($t1)						IF	ID	EX	MEM	WB							
Εντολή 7	ADD $t2, $t6, $t5							IF	ID	EX	MEM	WB						
Εντολή 8	SW $t2, 8($t1)								IF	ID	EX	MEM	WB					
Εντολή 9	ADDI $t4, $t4, -4									IF	ID	EX	MEM	WB				
Εντολή 10	BNE $t9, $t4, LOOP										IF	ID	EX	MEM	WB			
Εντολή 11	LW $t2, 8($t1)													IF	ID	EX	MEM	WB

Οι πίνακες έγιναν στο Excel

Για κάθε ζεύγος που εμφανίζεται RAW εξάρτηση σημειώσαμε είτε με χρώμα, υπογράμμιση ή τετραγωνάκι.

Το τελικό διάγραμμα χρονισμού είναι το παρακάτω:

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20	CC21	CC22	CC23	CC24	CC25	CC26	CC27	CC28	CC29
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM	WB																								
Εντολή 2	ADD $t2, $t2, $t1		IF	ID	-	-	EX	MEM	WB																					
Εντολή 3	SW $t2, 8($t1)			IF	-	-	ID	-	-	EX	MEM	WB																		
Εντολή 4	ADDI $t1, $t1, 8						IF	-	-	ID	EX	MEM	WB																	
Εντολή 5	LW $t6, 8($t1)									IF	ID	-	-	EX	MEM	WB														
Εντολή 6	LW $t5, 16($t1)										IF	-	-	ID	EX	MEM	WB													
Εντολή 7	ADD $t2, $t6, $t5													IF	ID	-	-	EX	MEM	WB										
Εντολή 8	SW $t2, 8($t1)														IF	-	-	ID	-	-	EX	MEM	WB							
Εντολή 9	ADDI $t4, $t4, -4																	IF	-	-	ID	-	-	EX	MEM	WB				
Εντολή 10	BNE $t9, $t4, LOOP																				IF	-	-	ID	EX	MEM	WB			
Εντολή 11	LW $t2, 8($t1)																									IF	ID	EX	MEM	WB

Με τις παύλες συμβολίζουμε τα stalls.

5) ΥΠΟΕΡΩΤΗΜΑ 1 ΚΥΚΛΟΙ ΓΙΑ ΟΛΕΣ ΤΙΣ ΕΠΑΝΑΛΗΨΕΙΣ

Για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η):

Από τον δοσμένο κώδικα παρατηρούμε ότι σε κάθε κύκλο $t9=$t9=0x1000=$16^3 = (4096)_{10}$

$t4-=4=0x1400-4=$16^3 + 4 \cdot 162 - 4 = (5120)_{10} - 4 = (5116)_{10}$

Επομένως το LOOP θα τρέξει 5120-4096=1024 δηλαδή μέχρι το $t4 να μειωθεί κατά 1024 ή αφού σε κάθε επανάληψη μειώνεται κατά 4 θα τρέξει $\frac{1024}{4} = 256$ φορές

Αρκεί να μετατρέψουμε τον αριθμό φορών που θα τρέξει το LOOP σε αριθμό κύκλων ρολογιού που θα τρέξει το LOOP.

Επομένως το πρόγραμμα θα ολοκληρωθεί σε 255*24+26 = 6146 κύκλους του ρολογιού.

6) ΥΠΟΕΡΩΤΗΜΑ 2 ΕΚΦΩΝΗΣΗ

2) Υποθέστε τώρα ότι υπάρχουν όλα τα δυνατά σχήματα προώθησης. Δείξτε όπως και πριν το διάγραμμα χρονισμού για την 1η επανάληψη του βρόχου, υποδεικνύοντας τις προωθήσεις που γίνονται. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα;

7) ΥΠΟΕΡΩΤΗΜΑ 2 ΔΙΑΓΡΑΜΜΑ ΧΡΟΝΙΣΜΟΥ

Στους κύκλους 4 και 10 υπάρχει ακόμη stall.
Η εντολή LW $t2, 8($t1) θα έχει έτοιμο το περιεχόμενο του $t2 στο στάδιο MEM του 4ου κύκλου οπότε χρειαζόμαστε ένα stall για να το προωθήσουμε στο στάδιο EX του 5ου κύκλου για την εντολή ADD $t2, $t2, $t1 όπου χρειάζεται να το χρησιμοποιήσουμε.
Ομοίως η εντολή LW $t5, 16($t1) θα έχει έτοιμο το περιεχόμενο του $t5 στο στάδιο MEM του 10ου κύκλου οπότε χρειαζόμαστε ένα stall για να το προωθήσουμε στο στάδιο EX του 11ου κύκλου για την εντολή ADD $t2, $t6, $t5 όπου χρειάζεται να το χρησιμοποιήσουμε.

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM	WB														
Εντολή 2	ADD $t2, $t2, $t1		IF	ID	-	EX	MEM	WB												
Εντολή 3	SW $t2, 8($t1)			IF	-	ID	EX	MEM	WB											
Εντολή 4	ADDI $t1, $t1, 8					IF	ID	EX	MEM	WB										
Εντολή 5	LW $t6, 8($t1)						IF	ID	EX	MEM	WB									
Εντολή 6	LW $t5, 16($t1)							IF	ID	EX	MEM	WB								
Εντολή 7	ADD $t2, $t6, $t5								IF	ID	-	EX	MEM	WB						
Εντολή 8	SW $t2, 8($t1)									IF	-	ID	EX	MEM	WB					
Εντολή 9	ADDI $t4, $t4, -4											IF	ID	EX	MEM	WB				
Εντολή 10	BNE $t9, $t4, LOOP												IF	ID	EX	MEM	WB			
Εντολή 11	LW $t2, 8($t1)															IF	ID	EX	MEM	WB

Με τα βελάκια συμβολίζουμε τα forwardings.

8) ΥΠΟΕΡΩΤΗΜΑ 2 ΚΥΚΛΟΙ ΠΟΥ ΑΠΑΙΤΟΥΝΤΑΙ ΤΩΡΑ

Για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η) θα τρέξει όπως και πριν 256 φορές

Επομένως το πρόγραμμα θα ολοκληρωθεί σε 255*14+16 = 3586 κύκλους του ρολογιού.

9) ΥΠΟΕΡΩΤΗΜΑ 3 ΕΚΦΩΝΗΣΗ

3) Θεωρώντας την ίδια σωλήνωση με το ερώτημα 2, μπορείτε να βελτιστοποιήσετε την επίδοση αναδιατάσσοντας τον κώδικα (με τις απαραίτητες βέβαια μετατροπές για να μην αλλάξετε την σημασιολογία του προγράμματος); Δείξτε όπως και πριν το διάγραμμα χρονισμού για την 1η επανάληψη του βρόχου, υποδεικνύοντας τις προωθήσεις που γίνονται. Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του βρόχου;

10) ΥΠΟΕΡΩΤΗΜΑ 3 ΔΙΑΓΡΑΜΜΑ ΧΡΟΝΙΣΜΟΥ

Βασιζόμενοι στο υποερώτημα 2 ψάχνουμε τρόπους αναδιάταξης του κώδικα για να αποφύγουμε τα δύο stalls που χρησιμοποιήσαμε.

Ψάχνουμε μία εντολή που βάζοντάς την ανάμεσα στην LW $t2, 8($t1) και στην ADD $t2, $t2, $t1 δεν αλλάζει η σημασιολογία του κώδικα και δεν δημιουργείται καινούργια εξάρτηση RAW.

Το ίδιο για την περίπτωση των εντολών LW $t5, 16($t1) και ADD $t2, $t6, $t5.

ΕΠΟΜΕΝΩΣ ΤΟ ΔΙΑΓΡΑΜΜΑ ΕΙΝΑΙ

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM	WB												
Εντολή 4	ADDI $t1, $t1, 8		IF	ID	EX	MEM	WB											
Εντολή 2	ADD $t2, $t2, $t1			IF	ID	EX	MEM	WB										
Εντολή 3	SW $t2, 8($t1)				IF	ID	EX	MEM	WB									
Εντολή 5	LW $t6, 8($t1)					IF	ID	EX	MEM	WB								
Εντολή 6	LW $t5, 16($t1)						IF	ID	EX	MEM	WB							
Εντολή 9	ADDI $t4, $t4, -4							IF	ID	EX	MEM	WB						
Εντολή 7	ADD $t2, $t6, $t5								IF	ID	EX	MEM	WB					
Εντολή 8	SW $t2, 8($t1)									IF	ID	EX	MEM	WB				
Εντολή 10	BNE $t9, $t4, LOOP										IF	ID	EX	MEM	WB			
Εντολή 11	LW $t2, 8($t1)													IF	ID	EX	MEM	WB

Με τα βελάκια συμβολίζουμε τα forwardings.

11) ΥΠΟΕΡΩΤΗΜΑ 3 ΚΥΚΛΟΙ ΠΟΥ ΑΠΑΙΤΟΥΝΤΑΙ ΤΩΡΑ

Για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η) θα τρέξει όπως και πριν 256 φορές

Επομένως το πρόγραμμα θα ολοκληρωθεί σε $255*12+14 = 3074$ κύκλους του ρολογιού.

12) ΥΠΟΕΡΩΤΗΜΑ 4 ΕΚΦΩΝΗΣΗ

4) Σε μια προσπάθεια βελτίωσης της επίδοσης του επεξεργαστή, σας προτείνουν το σπάσιμο του σταδίου MEM σε δύο ίσης διάρκειας στάδια, ώστε να μειωθεί η διάρκεια του κύκλου. Εκτελέστε τον αρχικό κώδικα στη καινούρια σωλήνωση των 6 σταδίων υποθέτοντας πως υπάρχουν όλα τα δυνατά σχήματα προώθησης. Δείξτε όπως και πριν το διάγραμμα χρονισμού για την 1η επανάληψη του βρόχου, υποδεικνύοντας τις προωθήσεις που γίνονται; Πόσοι κύκλοι απαιτούνται τώρα για την εκτέλεση του κώδικα; Βελτιώθηκε η επίδοση του επεξεργαστή;

13) ΥΠΟΕΡΩΤΗΜΑ 4 ΔΙΑΓΡΑΜΜΑ ΧΡΟΝΙΣΜΟΥ

Σπάμε το στάδιο MEM σε δύο στάδια ίσης διάρκειας MEM1=MEMWRITE και MEM2=MEMREAD. Έτσι η LW έχει έτοιμη από το στάδιο MEM1 του 4ου κύκλου την τιμή του $t2 που την προωθεί στο EX στάδιο του 5ου κύκλου. Ομοίως, η LW έχει έτοιμη από το στάδιο MEM1 του 10ου κύκλου την τιμή του $t5 που την προωθεί στο EX στάδιο του 11ου κύκλου.

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18	CC19	CC20
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM1	MEM2	WB														
Εντολή 2	ADD $t2, $t2, $t1		IF	ID	-	EX	MEM1	MEM2	WB												
Εντολή 3	SW $t2, 8($t1)			IF	-	ID	EX	MEM1	MEM2	WB											
Εντολή 4	ADDI $t1, $t1, 8					IF	ID	EX	MEM1	MEM2	WB										
Εντολή 5	LW $t6, 8($t1)						IF	ID	EX	MEM1	MEM2	WB									
Εντολή 6	LW $t5, 16($t1)							IF	ID	EX	MEM1	MEM2	WB								
Εντολή 7	ADD $t2, $t6, $t5								IF	ID	-	EX	MEM1	MEM2	WB						
Εντολή 8	SW $t2, 8($t1)									IF	-	ID	EX	MEM1	MEM2	WB					
Εντολή 9	ADDI $t4, $t4, -4											IF	ID	EX	MEM1	MEM2	WB				
Εντολή 10	BNE $t9, $t4, LOOP												IF	ID	EX	MEM1	MEM2	WB			
Εντολή 11	LW $t2, 8($t1)															IF	ID	EX	MEM1	MEM2	WB

Με τα βελάκια συμβολίζουμε τα forwardings.

14) ΥΠΟΕΡΩΤΗΜΑ 4 ΚΥΚΛΟΙ ΠΟΥ ΑΠΑΙΤΟΥΝΤΑΙ ΤΩΡΑ

Για να ολοκληρωθεί ο παραπάνω βρόχος (για όλες τις επαναλήψεις του, όχι μόνο για την 1η) θα τρέξει όπως και πριν 256 φορές

Επομένως το πρόγραμμα θα ολοκληρωθεί σε 255*14+17 = 3587 κύκλους του ρολογιού.

15) ΥΠΟΕΡΩΤΗΜΑ 4 ΣΧΟΛΙΑ ΓΙΑ ΤΟ ΣΠΑΣΙΜΟ

Ό, τι αναφέραμε παραπάνω αφορά ένα πολύ συγκεκριμένο σπάσιμο του σταδίου MEM στο οποίο θεωρούμε ότι στο πρώτο μισό MEM1 προλαβαίνουμε να γράψουμε MEWRITE και στο δεύτερο μισό MEM2 προλαβαίνουμε να διαβάσουμε MEMREAD. Η εκφώνηση αναφέρει ότι MEM1=MEM2 και τηρώντας την σύμβαση *« Τέλος, υποθέστε ότι η εγγραφή σε ένα καταχωρητή γίνεται στο πρώτο μισό ενός κύκλου, ενώ η ανάγνωση από τον ίδιο καταχωρητή πραγματοποιείται στο δεύτερο μισό του κύκλου.»* Επομένως οι υποθέσεις που κάναμε για το σπάσιμό μας MEM1≡MEMWRITE και MEM2≡MEMREAD και $t_{MEM1} \equiv t_{MEMWRITE} = t_{MEM2} \equiv t_{MEMREAD} = \frac{t_{MEM}}{2}$ είναι εύλογες.

16) ΥΠΟΕΡΩΤΗΜΑ 4 ΕΠΙΔΟΣΗ ΕΠΕΞΕΡΓΑΣΤΗ

Η επίδοση του επεξεργαστή με το σπάσιμο του σταδίου MEM σε δύο ίσης διάρκειας στάδια ώστε να μειωθεί η διάρκεια του κύκλου είναι μια συμφέρουσα ιδέα μόνο αν τα στάδια διαρκούν διαφορετικούς χρόνους. Κάθε κύκλος θα έχει την διάρκεια του σταδίου της χειρότερης περίπτωσης και για αυτό αν το MEM ήταν το στάδιο της χειρότερης περίπτωσης (χρονικά) πράγματι θα βελτιωνόταν αρκεί ο χρόνος που γλιτώνουμε να είναι περισσότερος από τον χρόνο που χάνουμε δεδομένου ότι με αυτή την υλοποίηση πρέπει να τρέξουμε έναν παραπάνω κύκλο ρολογιού.

Στην περίπτωσή μας θεωρούμε ότι *«Όλα τα στάδια διαρκούν ένα κύκλο.»* επομένως με το να σπάσουμε το στάδιο MEM στα δύο δεν μειώνεται ο χρόνος διάρκειας του ενός κύκλου ρολογιού. Ο κύκλος ρολογιού παραμένει ίδιος και το πρόγραμμα τρέχει για περισσότερους κύκλους (σωλήνωση 6 σταδίων με όλα τα δυνατά σχήματα προώθησης 3587) σε σχέση με το πρόγραμμα του υποερωτήματος 2 (σωλήνωση 5 σταδίων με όλα τα δυνατά σχήματα προώθησης 3586) για τον ίδιο χρόνο ανά κύκλο επομένως η επίδοση του επεξεργαστή μειώθηκε.

17) ΥΠΟΕΡΩΤΗΜΑ 5 ΕΚΦΩΝΗΣΗ

5) Δίνεται πως στο αρχικό datapath η διάρκεια του κάθε σταδίου ήταν η εξής:

IF: 240ps, ID: 120ps, EX: Tps, MEM: 400ps, WB: 120ps

Υπολογίστε τη μέγιστη διάρκεια του σταδίου EX ώστε η βέλτιστη επίδοση να επιτυγχάνεται με την αλλαγή του υλικού *(ερώτημα 4)* και όχι με την αλλαγή του κώδικα *(ερώτημα 3)*.

Θα έχω 3074 κύκλους των 400ps δηλαδή 3074*400=1,229,600 s = 205 h η συνολική διάρκεια του προγράμματος με αλλαγή κώδικα σε σωλήνωση 5 σταδίων.

Σε αυτό το υποερώτημα υπάρχει ουσιαστική αλλαγή με το σπάσιμο του σταδίου MEM σε δύο ίσης διάρκειας στάδια.

Αρχικά δοκιμάζουμε με την διάρκεια του σταδίου EX να είναι αυτή της δεύτερης χειρότερης περίπτωσης δηλαδή EX:240ps (όσο δηλαδή του IF). Το MEM που είναι η χειρότερη περίπτωση θα σπάσει στην μέση (MEM1=MEMWRITE:200ps και MEM2=MEMREAD:200ps).

Θα έχω 3587 κύκλους των 240ps δηλαδή 3587*240=860,880 s η συνολική διάρκεια του προγράμματος με αλλαγή υλικού.

Όμως 1229,600 s = 205 h<<860,880 s

Επομένως για να βρούμε την μέγιστη τιμή θα επιλέξουμε μια τιμή EX>240ps αλλάζοντας έτσι τον κύκλο του ρολογιού.

Θέλουμε $3587 * EX \leq$ 1,229,600 ή $EX \leq 342.79$ δηλαδή $EX_{max} = 342$ ps

3587*342=1226754 s < 1,229,600 s πράγματι βελτιώσαμε την επίδοση με αλλαγή υλικού.

18) ΥΠΟΕΡΩΤΗΜΑ 6 ΕΚΦΩΝΗΣΗ

6) Μπορείτε να βελτιστοποιήσετε και την επίδοση της σωλήνωσης του ερωτήματος 4 αναδιατάσσοντας τον κώδικα; Ποια είναι η μέγιστη διάρκεια του σταδίου EX ώστε η βέλτιστη επίδοση να επιτυγχάνεται σε αυτή την περίπτωση (δηλ. με αλλαγή υλικού & αναδιάταξη κώδικα);

	Κύκλος	CC1	CC2	CC3	CC4	CC5	CC6	CC7	CC8	CC9	CC10	CC11	CC12	CC13	CC14	CC15	CC16	CC17	CC18
Εντολή 1	LW $t2, 8($t1)	IF	ID	EX	MEM1	MEM2	WB												
Εντολή 4	ADDI $t1, $t1, 8		IF	ID	EX	MEM1	MEM2	WB											
Εντολή 2	ADD $t2, $t2, $t1			IF	ID	EX	MEM1	MEM2	WB										
Εντολή 3	SW $t2, 8($t1)				IF	ID	EX	MEM1	MEM2	WB									
Εντολή 5	LW $t6, 8($t1)					IF	ID	EX	MEM1	MEM2	WB								
Εντολή 6	LW $t5, 16($t1)						IF	ID	EX	MEM1	MEM2	WB							
Εντολή 9	ADDI $t4, $t4, -4							IF	ID	EX	MEM1	MEM2	WB						
Εντολή 7	ADD $t2, $t6, $t5								IF	ID	EX	MEM1	MEM2	WB					
Εντολή 8	SW $t2, 8($t1)									IF	ID	EX	MEM1	MEM2	WB				
Εντολή 10	BNE $t9, $t4, LOOP										IF	ID	EX	MEM1	MEM2	WB			
Εντολή 11	LW $t2, 8($t1)													IF	ID	EX	MEM1	MEM2	WB

Με αυτή την υλοποίηση (αλλαγή κώδικα και αλλαγή υλικού σωλήνωση 6 σταδίων) έχω 255*12+15 =3075. Θα έχω 3075 κύκλους των EXps.

Με αλλαγή κώδικα σε σωλήνωση 5 σταδίων: Θα έχω 3074 κύκλους των 400ps δηλαδή 3074*400=1,229,600 s = 205 h η συνολική διάρκεια του προγράμματος.

Θέλουμε $3074 * EX \leq 1{,}229{,}600$ ή $EX \leq 400$ δηλαδή $EX_{max} = 399$ ps

3074*399=1226526 s < 1,229,600 s πράγματι βελτιώσαμε την επίδοση με την ζητούμενη υλοποίηση.