1) <u>Υλοποίηση εντολών του επεξεργαστή MIPS</u>

Η εντολή MIPS *`jalr rs`* χρησιμοποιείται μερικές φορές αντί της *`jal`* για την κλήση συναρτήσεων. Η εντολή αυτή ακολουθεί το R-format και ο καταχωρητής *`rs`* κωδικοποιείται στα bits [25-21] της εντολής.

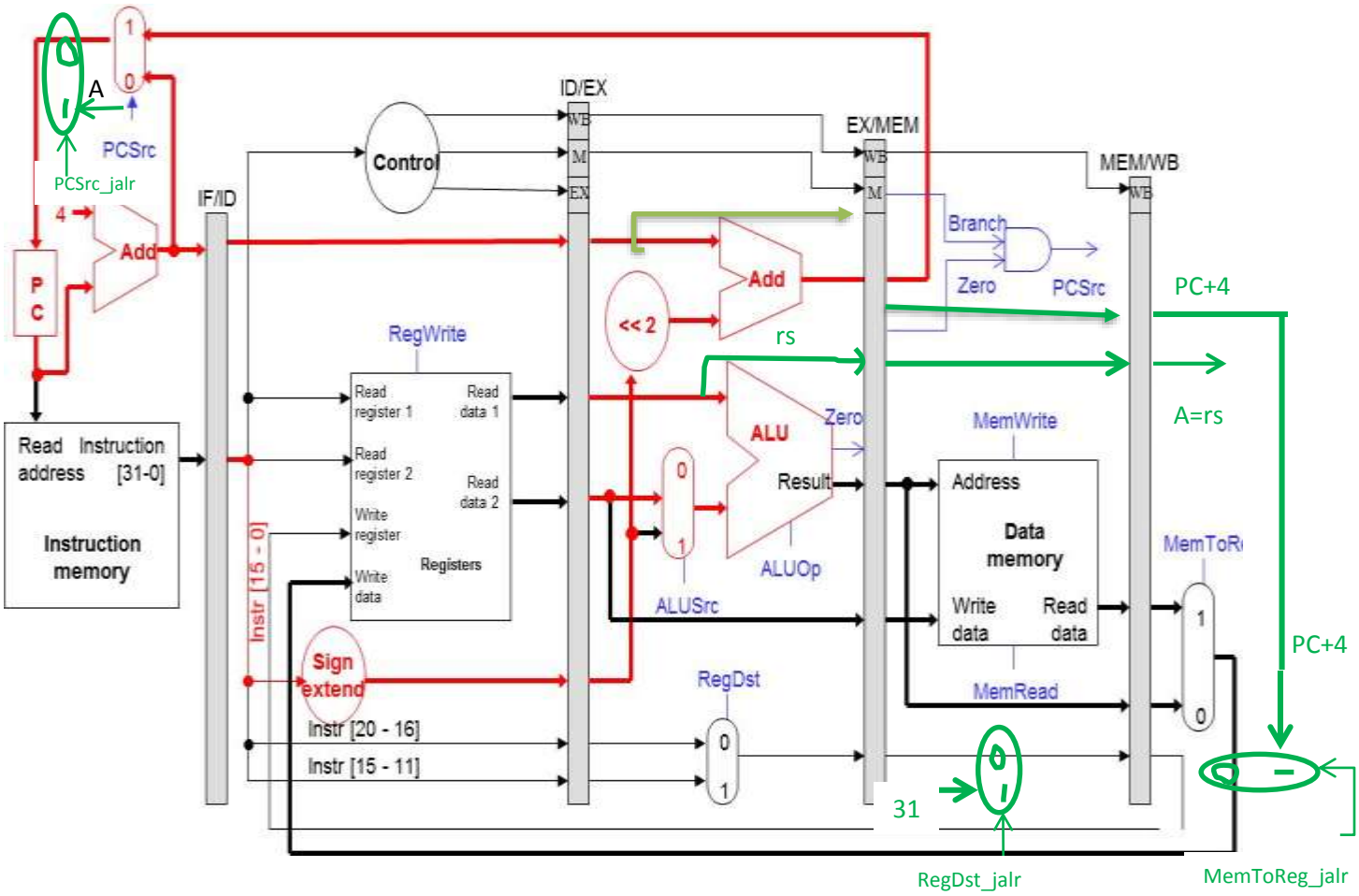
6	5	5	5	5	6
Op= 000000	rs	X	X	00000	Func=001001

Η άσκηση αυτή σας ζητάει να υλοποιήσετε την εντολή αυτή ξεκινώντας από την βασική αρχιτεκτονική διοχέτευσης του MIPS που σας δίδεται στην επόμενη σελίδα του εντύπου. Όλες οι αλλαγές θα πρέπει να γίνουν πάνω στο σχήμα που σας δίδεται και θα πρέπει να είναι καθαρογραμμένες! Πιο συγκεκριμένα θα πρέπει:

α) να υλοποιήσετε όλες τις αλλαγές στο τμήμα δεδομένων. Μπορείτε να προσθέσετε καινούργια modules (πχ πολυπλέκτες, ALUs, κοκ), αλλά δεν επιτρέπεται να αλλάξετε την λειτουργικότητα ή το μέγεθος κανενός ήδη υπάρχοντος module. Επίσης μπορείτε να προσθέσετε καινούργια σήματα και συνδέσεις. Αγνοείστε τυχόν θέματα προώθησης (forwarding) της *`jalr`* σε σχέση με προηγούμενες εντολές.

β) όλες τις αλλαγές στο τμήμα ελέγχου. Αυτό συμπεριλαμβάνει τις τιμές τυχόν νέων σημάτων ελέγχου που θα δημιουργηθούν για τις ανάγκες της εντολής αυτής και για τις παλιές εντολές, καθώς και τις τιμές των ήδη υπαρχόντων σημάτων ελέγχου που φαίνονται στο σχήμα για την *`jalr`*. Στον παρακάτω πίνακα σας δίνονται οι τιμές των σημάτων ελέγχου για μερικές εντολές που εκτελεί ο επεξεργαστής και εσείς θα πρέπει να συμπληρώσετε τα λευκά κενά.

γ) Μπορείτε να επισημάνετε προβλήματα που θα υπάρξουν με την εντολή *`jalr rs`* σε σχέση με εντολές διακλάδωσης που μπορούν να βρίσκονται αμέσως μετά από την *`jalr`* στον κώδικα και οι οποίες εκτελούνται στο ID στάδιο; Αφού επισημάνετε το πρόβλημα, μπορείτε να δώσετε κάποια λύση για το πώς μπορεί να επιλυθεί;



Εντολή	RegWrite	RegDst	ALUSrc	Branch	MemWrite	MemToReg	ALUOp [1:0]	PCSrc	PCSrc_jalr	RegDst_jalr	MemToReg _jalr
R-format	1	1	0	0	0	0	10	0	0	0	0
lw	1	1	1	0	0	1	00 (add)	0	0	0	0
sw	0	X	1	0	1	X	00 (add)	0	0	0	0
beq	0	X	0	1	0	X	01 (sub)	1	0	0	0
jalr	1	X	X	X	0	X	X	X	1	1	1

Λύση

a) Η εντολή *jalr rs* εκτελεί την παρακάτω ακολουθία εντολών: *$ra = PC+4, PC = rs*.

Στο παραπάνω σχήμα φαίνονται και οι αλλαγές που πρέπει να γίνουν στο τμήμα δεδομένων. Σημειώστε ότι επειδή πρέπει να γράψουμε τον καταχωρητή $ra στο στάδιο WB και αυτή η αλλαγή δεν θα πρέπει να γίνει μετά την αλλαγή *PC = rs,* θα πρέπει ο PC να αλλάξει και αυτός στο στάδιο WB. Οι επιπλέον προσθήκες φαίνονται σε πράσινο χρώμα.

β) Χρειάζονται τρία επιπλέον σήματα επιλογής εισόδου των τριών πολυπλεκτών που φαίνονται στο σχήμα.

γ) Έστω ότι έχουμε τον παρακάτω κώδικα:

```
  …..
        jalr $t0
L:      add  ….
        beqz $t1, $t2, L
```

Όταν η εντολή jalr βρίσκεται στο στάδιο MEM, η εντολή beqz θα βρίσκεται στο στάδιο ID και μπορεί να αλλάξει την ροή του προγράμματος εάν ($t1==$t2). Αυτό φυσικά θα είναι λάθος γιατί η beqz, όπως και η εντολή add δεν θα πρέπει να εκτελεσθούν εφόσον η jalr προηγείται.

Η λύση είναι να κάνουμε flush το pipeline όταν διαγνώσουμε την εντολή jalr στο στάδιο ID. Στην συνέχεια παγώνουμε το pipeline, (δηλ. εισάγουμε μόνο εντολές nops) μέχρι να υπολογισθεί η νέα διεύθυνση από την οποία θα φέρουμε δεδομένα (δηλ. η ($t0)) στο στάδιο WB.

2) <u>Αρχιτεκτονική Διοχέτευσης και Καθυστερήσεις</u>

Έστω το παρακάτω loop το οποίο χρησιμοποιείται για να διατρέξουμε μια δομή δεδομένων:

```
loop: lw  $t0, 0($a0)
      beq $a1, $t0, exit
      bge $a1, $t0, gt
      lw  $a0, 4($a0)
      j   end
gt:   lw  $a0, 8($a0)
end:  bne $a0, $0, loop
exit:
```

α) Κυκλώστε <u>όλες</u> τις εξαρτήσεις δεδομένων (data dependencies) μέσα σε μία επανάληψη του loop.

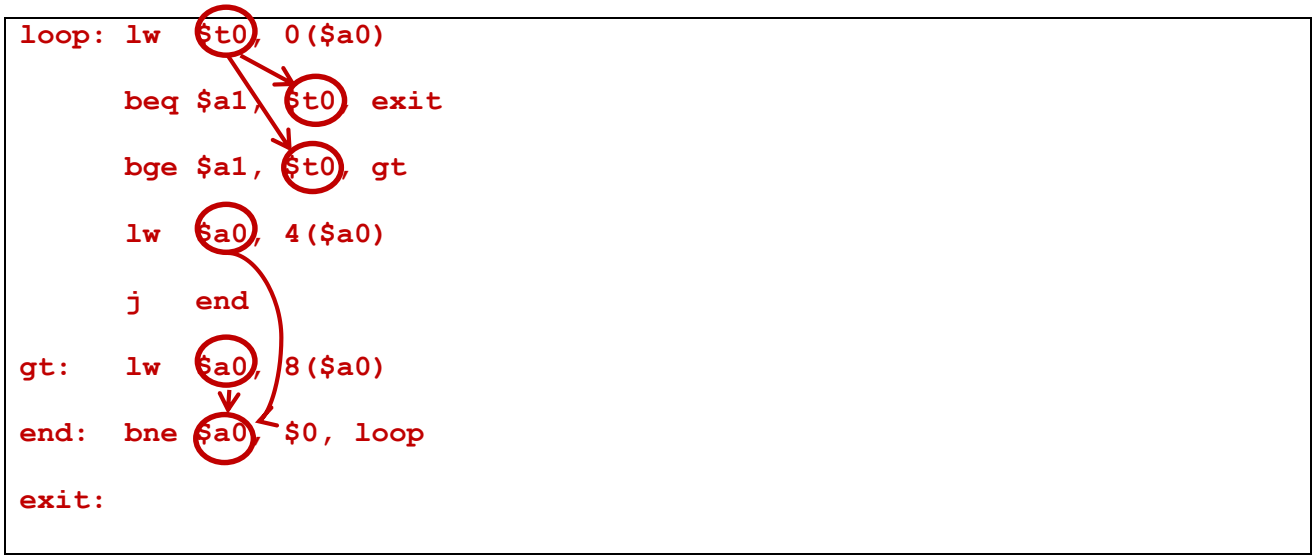
β) Θεωρούμε την αρχιτεκτονική διοχέτευσης των 5 σταδίων με πλήρη προώθηση. Οι εντολές διακλάδωσης είναι πάντα "Predict Not Taken" και η εκτέλεση τους γίνεται στο στάδιο MEM (και όχι στο ID). Συμπληρώστε το παρακάτω πίνακα θεωρώντας ότι η εντολή bge είναι πάντα Taken και ότι το loop εκτελείται πολλές φορές. Προσέξτε ότι οι εντολές στο παρακάτω πίνακα είναι με την σειρά με την οποία εκτελούνται.

Εντολή	Επανάληψη	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
lw	1	F	D	E	M	W																	
beq	1																						
bge	1																						
lw	1																						
bne	1																						
lw	2																						
beq	2																						
bge	2																						

γ) Υπολογίστε το μέσο CPI για κάθε επανάληψη του loop χρησιμοποιώντας την λύση του ερωτήματος β.

Λύση

α)



```
loop: lw   $t0, 0($a0)
      beq  $a1, $t0, exit
      bge  $a1, $t0, gt
      lw   $a0, 4($a0)
      j    end
gt:   lw   $a0, 8($a0)
end:  bne  $a0, $0, loop
exit:
```

β)

Εντολή	Επανάληψη	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2
lw	1	F	D	E	M	W																	
beq	1		F	D	D	E	M	W															
bge	1			F	F	D	E	M	W														
lw	1					X	X	X	F	D	E	M	W										
bne	1									F	D	D	E	M	W								
lw	2										X	X	X	X	F	D	E	M	W				
beq	2															F	D	D	E	M	W		
bge	2																F	F	D	E	M	W	

γ) Για την εκτέλεση ενός loop χρειαζόμαστε 18-5 = 13 κύκλους μηχανής: τέλος lw της 1^{ης} επανάληψης του loop στον κύκλο 5 μέχρι τέλος της lw της 2^{ης} επανάληψης του loop στον κύκλο 18. Άρα χρειαζόμαστε 13 κύκλους για 5 εντολές με CPI = 13/5 = 2,6.