ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
TECHNICAL UNIVERSITY OF CRETE

Οργάνωση Υπολογιστών

Εργαστηριακή Αναφορά Επεξεργαστή Pipeline

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Μιχαηλίδης Στέργιος 2020030080
Αγγελόπουλος Δημήτριος 2020030038

23.06.2024

Ερώτημα Α

Σκοπός της άσκησης είναι να μετατρέψουμε το DATAPATH της άσκησης 2 με κατάλληλο τρόπο ώστε να υλοποιεί έναν επεξεργαστή πολλαπλών κύκλων Pipeline. Αυτό το επιτυγχάνουμε κρατώντας κάθε σήμα ελέγχου σε ένα διάνυσμα και στην συνέχεια τροφοδοτώντας το σε μία σειρά καταχωρητών. Οι καταχωρητές αυτοί θα είναι 4, ένας για το DEC, ένας για το EXEC, ένας για το MEM και ένας για το WB όπως φαίνεται στο παρακάτω σχήμα.

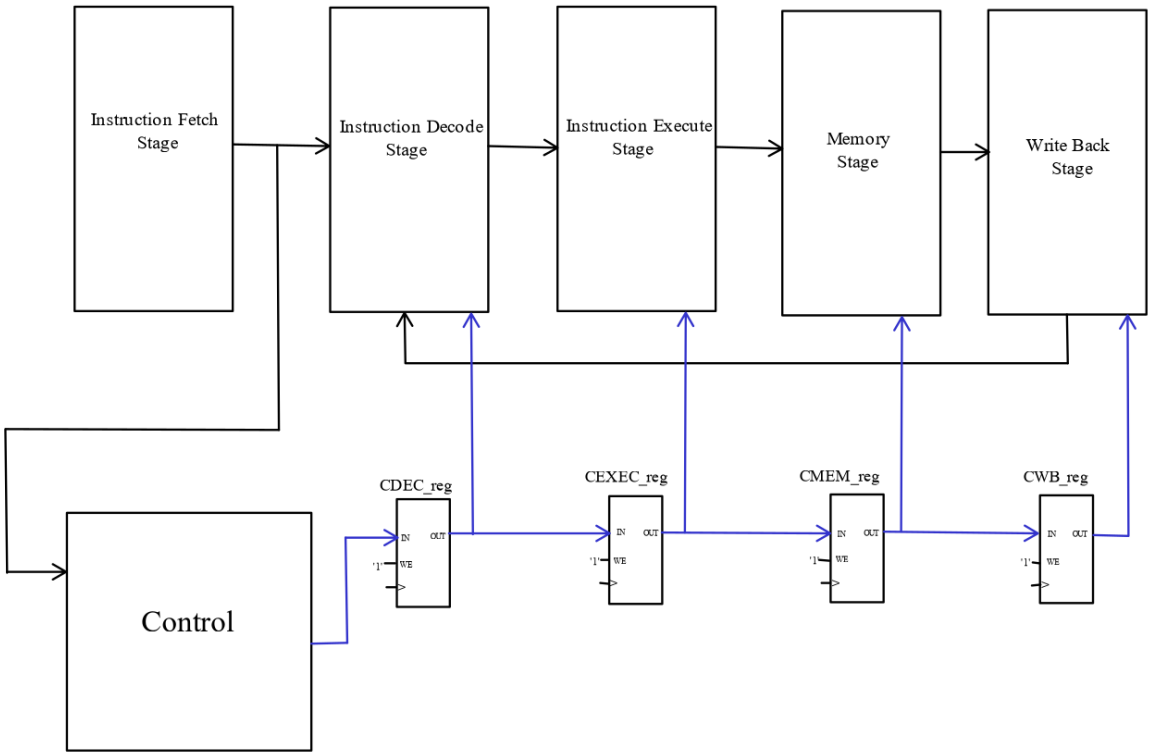

Figure 1: Pipelined Datapath

Με αυτήν την αλλαγή πετυχαίνουμε την λειτουργία πολλαπλών κύκλων Pipeline.

Ωστόσο, η λογική ενός pipelined επεξεργαστή δημιουργεί μερικά νέα προβλήματα, τα Data-Hazards. Ο τρόπος αντιμετώπισης αυτών γίνεται μέσω Forwarding ή/και μέσω Stalls. Εμείς δημιουργήσαμε ένα νέο Unit το οποίο υλοποιεί την λογική της προώθησης και αντιμετωπίζει τα Read-After-Write Hazards που προκύπτουν στην βαθμίδα EXEC. Για την υλοποίηση της λογικής αυτής προσθέσαμε και μερικούς επιπλέον πολυπλέκτες και καταχωρητές στο Datapath όπως φαίνεται παρακάτω.

Το νέο Unit που προσθέσαμε είναι το Forwarding Unit το οποίο είναι μία μονάδα ελέγχου η οποία έχει σκοπό να προωθήσει τα δεδομένα μερικών καταχωρητών όταν δεν έχει ολοκληρωθεί το Write Back μίας προηγούμενης εντολής λόγω του Pipeline.

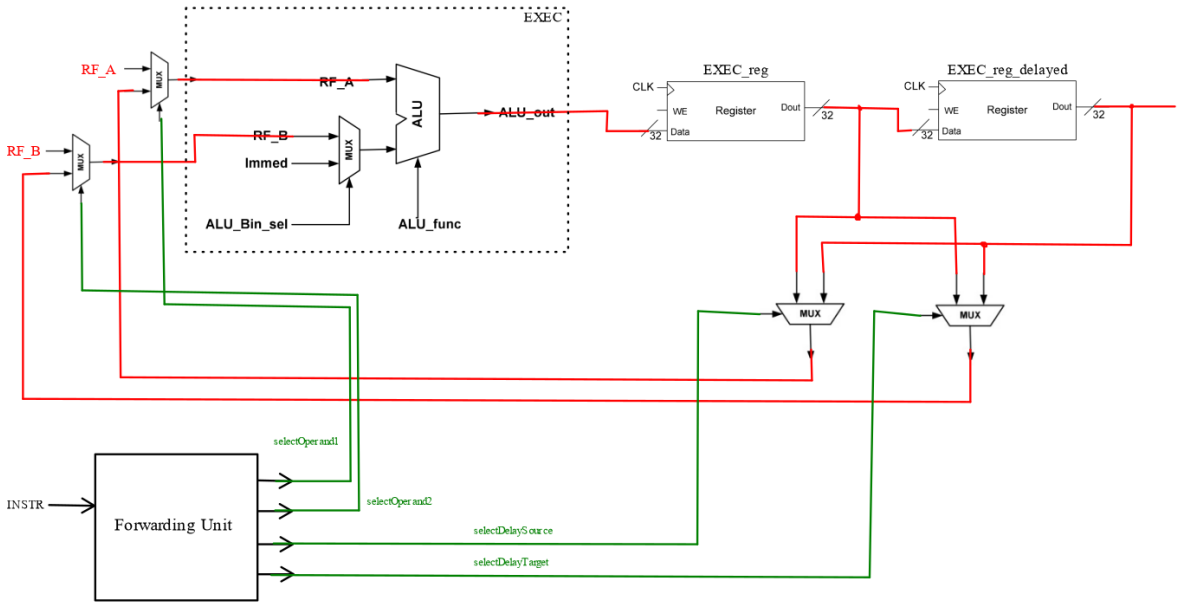

Figure 2: Forwarding Logic

Ερώτημα Β

Σε αυτό το ερώτημα θα κάνουμε τις απαραίτητες αλλαγές στο Control του επεξεργαστή. Αρχικά θέλουμε το Control Unit να μας δίνει μία έξοδο-δίαυλο ως εξής :

$$\begin{aligned} CONTROL\_Out = \; & [MEMreg\_WrEn, EXECreg\_WrEn, DECreg\_WrEn, IFreg\_WrEn, \\ & ByteFlag, MEM\_WrEn, ALU\_func, ALU\_Bin\_sel, RF\_B\_sel, \\ & RF\_WrData\_sel, RF\_WrEn, PC\_LdEn, PC\_sel] \end{aligned}$$

Αυτή η έξοδος αποτελείται από 16bits. Επομένως την κάνουμε Zero-fill έτσι ώστε να μπορούμε να την τροφοδοτίσουμε στον πρώτο Pipeline register ο οποίος είναι των 32bits.

Στην συνέχεια υλοποιούμε την λογική του Forwarding Unit. Η λογική αυτή διαφοροποιείται μεταξύ των R-Type και I-Type(μη λαμβάνοντας υπόψη τα branches και τις sw,lw,lb) εντολών. Παρατηρούμε ότι έχουμε RAW Hazard σε περίπτωση που κάποιος ζητούμενος Register βρισκεται ως Destination Register σε κάποια από τις δύο προηγούμενες εντολές. Επίσης αυτό ισχυεί μόνο για εντολές που έχουν διαφορά δύο κύκλων το πολύ μεταξύ τους.

1. R-Type Forwarding

   Σε αυτήν την περίπτωση θεωρούμε ότι η εντολή που βρίσκεται αυτήν την στιγμή στο DEC stage έχει σαν Rs ή/και Rt κάποιον Register ο οποίος ήταν ο Rd σε κάποια προηγούμενη εντολή. Λόγω του Pipeline κάποιος από τους δύο πρώτους μπορεί να μην έχει φτάσει στο στάδιο του WB ώστε να μπορούμε χρησιμοποιήσουμε το περιεχόμενο του στο EXEC της εντολής της οποία θέλουμε να εκτλέσουμε μια πράξη. Για να διορθώσουμε το Hazard συγκρίνουμε τους Rs, Rt με τους Rd των δύο προηγούμενων εντολών. Αρχικά ξεχωριστά τον καθένα για κάθε εντολή και τέλος συγκρίνουμε αν πρέπει να κάνουμε διπλή προώθηση, δηλαδή αν και οι δυο Rs, Rt ήταν destinations των δύο προηγούμενων εντολών.

2. I-Type Forwarding

   Σε αυτήν την περίπτωση έχουμε μόνο τον Rs και επομένως ο έλεγχος είναι πολύ πιο απλός. Εδώ αρκεί να δούμε αν ο Rs έχει ίδια διεύθυνση με τον Rd της εντολής του ενός κύκλου πριν ή με τον Rd της εντολής των δύο κύκλων πριν και κάνουμε προώθηση το αντίστοιχο αποτέλεσμα της ALU.