26/5/2019

5ο Εργαστήριο Οργάνωσης Υπολογιστών
ΜΕΤΑΤΡΟΠΗ ΤΟΥ CHARIS-4 ΣΕ PIPELINE PROCESSOR
10/5/2019

Ομάδα LAB31239632

Αλέξανδρος Πουπάκης 2016030061
Βαγγέλης Κιούλος 2016030056

Σκοπός εργαστηριακής άσκησης

Στο εργαστήριο αυτό έπρεπε να μετατρέψουμε τον multicycle επεξεργαστή του προηγούμενου εργαστηρίου σε pipelined επεξεργαστή που να υποστηρίζει 4 εντολές.

Προεργασία

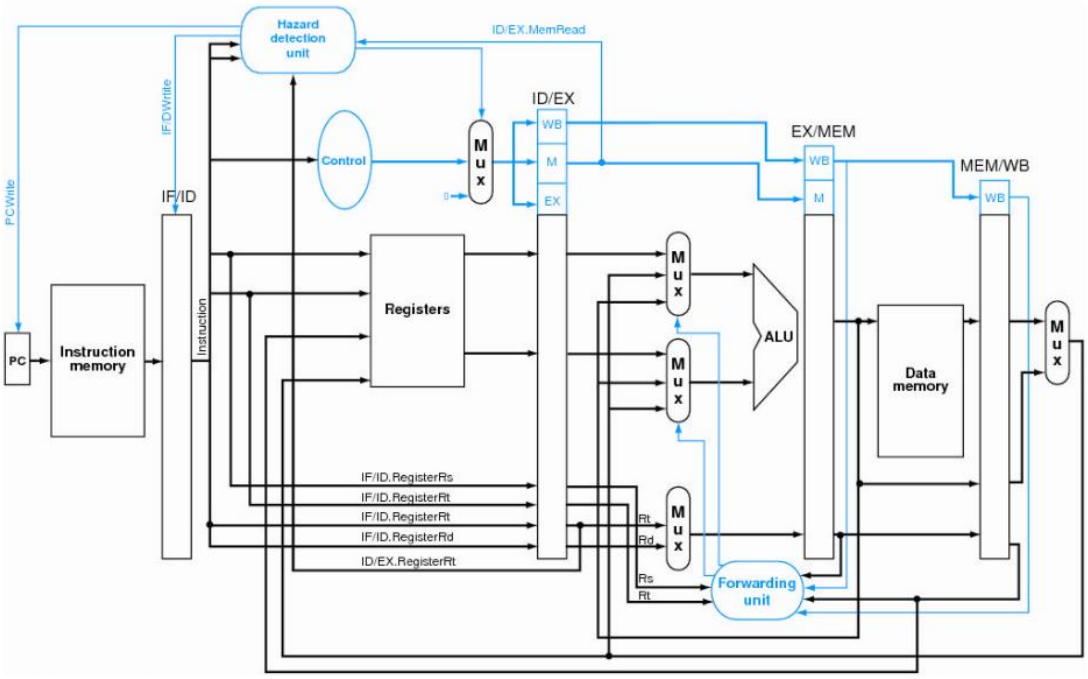
Η εντολή addi παράγει το ίδιο αποτέλεσμα με την li, οπότε για την υλοποίηση της δεύτερης χρησιμοποιήθηκε η πρώτη χωρίς κάποια επιβάρυνση, αφού στον pipelined επεξεργαστή κάθε εντολή «περνάει» μέσα από όλες τις βαθμίδες. Για τις εντολές add, li, lw, sw, παράχθηκε ο παρακάτω πίνακας με τις τιμές των σημάτων ελέγχου (προκύπτει από τον αντίστοιχο πίνακα του προηγούμενου εργαστηρίου).

	add	li (addi)	lw	sw
PC_sel	0	0	0	0
PC_LdEn	1	1	1	1
RF_WrEn	1	1	1	0
RF_WrData_sel	0	0	1	**d**
RF_B_sel	0	1	**d**	1
ImmExt_s	**d**	1	1	1
ALU_Bin_sel	0	1	1	1
ALU_func	func	110000	110000	110000
Mem_WrEn	0	0	0	1

Η σχεδίαση που υλοποιεί τον ζητούμενο επεξεργαστή φαίνεται στο παρακάτω σχήμα που παρουσιάστηκε στις διαλέξεις του μαθήματος.

Υπάρχουν τρία κύρια σημεία που πρέπει να ληφθούν υπόψιν για τη σωστή σχεδίαση του επεξεργαστή:

- Το control, το forwarding unit και το hazard detection unit είναι συνδυαστικά κυκλώματα διότι στον ίδιο κύκλο που τους «έρχονται» τα σήματα εισόδου, το υπόλοιπο κύκλωμα χρειάζεται τα σήματα εξόδου. Άρα τα modules αυτά δεν μπορούν να είναι clocked.
- Hazard έχουμε μόνο στην εντολή load και σε αυτή την περίπτωση εισάγουμε καθυστέρηση ενός κύκλου (no-op) στο datapath.
- Forwarding κάνουμε όταν μια εντολή χρησιμοποιεί στην είσοδο έναν καταχωρητή που ενημερώθηκε σε προηγούμενη εντολή και δεν έχει ακόμα γραφτεί στην Register File



Περιγραφή

Για την υλοποίηση του παραπάνω κυκλώματος, αρχικά τροποποιήθηκε το control ώστε να υπολογίζει όλα τα σήματα ελέγχου για μια εντολή στον ίδιο κύκλο που άλλαξε η εντολή. Δηλαδή από FSM που ήταν το control στο προηγούμενο εργαστήριο, μετατράπηκε σε ένα απλό συνδυαστικό κύκλωμα.

Στη συνέχεια προστέθηκαν καταχωρητές στα επίπεδα Decode, ALU και Memory οι οποίοι αποθηκεύουν τα απαραίτητα σήματα ελέγχου καθώς και τη διεύθυνση των καταχωρητών rt και rs, σύμφωνα με το παραπάνω σχήμα. Προστέθηκαν επίσης και πολυπλέκτες για την επιλογή των σημάτων ελέγχου (control signals ή no-op) και των τελεστέων (δεδομένα από την RF ή forwarded δεδομένα).

Έπειτα υλοποιήθηκε το Forwarding Unit σύμφωνα με τις συνθήκες που παρουσιάστηκαν στις διαλέξεις, δηλαδή γίνεται forwarding όταν μια νέα εντολή χρειάζεται αποτέλεσμα κάποιας προηγούμενης (και το αποτέλεσμα δεν έχει γραφτεί ακόμα στην RF), η οποία γράφει τους καταχωρητές και ο rd της δεν είναι ο zero register.

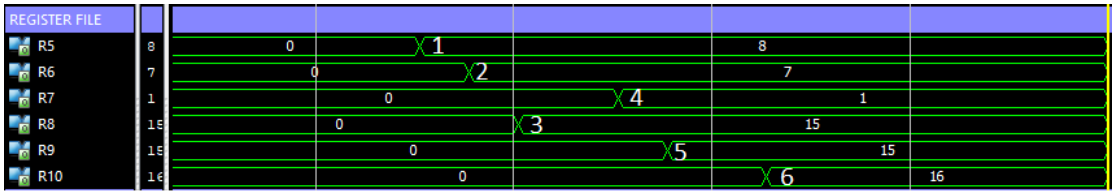
Τέλος, υλοποιήθηκε το Hazard Detection Unit, το οποίο αποτρέπει τη χρήση του καταχωρητή rd μιας εντολής lw, αμέσως μετά την εκτέλεσή της. Σε τέτοια περίπτωση, ο πολυπλέκτης βγάζει στην έξοδο μηδενικά για όλα τα σήματα ελέγχου και με τον τρόπο αυτό καθυστερεί η εκτέλεση της νέας εντολής κατά ένα κύκλο.

Κυματομορφές - Προσομοίωση

Για την δοκιμή του pipeline, εκτελείται το παρακάτω ενδεικτικό πρόγραμμα:

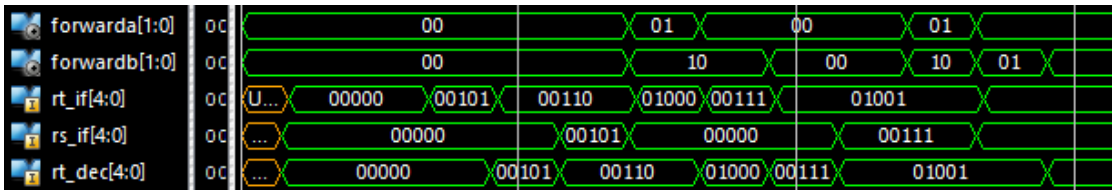
```
1   li r5, 8
2   li r6, 7
3   add r8, r5, r6
4   sw r8, 4(r0)
5   li r7, 1
6   lw r9, 4(r0)
7   add r10, r9, r7
8
```

Κυματομορφές προσομοίωσης προγράμματος:



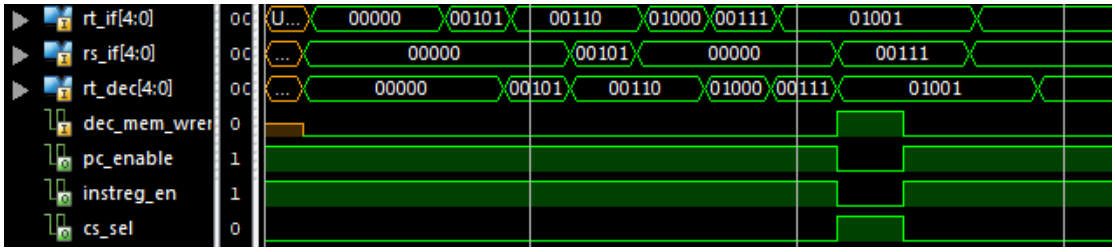
Παραπάνω, βλέπουμε το αποτέλεσμα από την εκτέλεση του προγράμματος μας. Αρχικά, παρατηρούμε ότι μετά την εκτέλεση της εντολής “li r5, 8”(1), στον καταχωρητή r5, βρίσκεται η τιμή “8”. Στην συνεχεία, με την εντολή “li r6, 7”(2), η τιμή 7 έχει καταχωρηθεί στον καταχωρητή r6. Έπειτα, εκτελώντας την εντολή “add r8, r5, r6”(3) προσθέτουμε τις τιμές των 2 καταχωρητών και καταχωρούμε το αποτέλεσμα (δηλ. το 15) στον καταχωρητή r8. Με την εντολή “sw r8, 4(r0)” αποθηκεύουμε στην μνήμη τα περιεχόμενα του καταχωρητή r8 και με την εντολή “li r7, 1”(4), βάζουμε στον καταχωρητή r7 την τιμή 1. Στο τέλος, με την εντολή “lw r9, 4(r0)”(5) φορτώνουμε στον καταχωρητή r9 την τιμή 15, και με την εντολή “add r10, r9, r7”(6) προσθέτουμε τα περιεχόμενα των καταχωρητών r9 και r7 και καταχωρούμε το αποτέλεσμα (δηλ. το 16) στον καταχωρητή r10.

Forwarding:



Παραπάνω παρατηρούμε την περίπτωση που χρειαζόμαστε forwarding για την συνεχεία της εκτέλεσης του προγράμματος. Παρατηρούμε ότι όταν το MEMtoWB_rd είναι ίσο με το DEC_rs τότε το forwardA είναι “01” ενώ όταν το ALUtoMEM_rd είναι ίσο με το DEC_rs τότε το forwardA είναι “10” και όταν το MEMtoWB_rd είναι ίσο με το DEC_rt τότε το forwardB είναι “01” ενώ όταν το ALUtoMEM_rd είναι ίσο με το DEC_rt τότε το forwardB είναι “10”.

Stall:



Παραπάνω, παρατηρούμε την περίπτωση στην οποία χρειάστηκε stall για την αντιμετώπιση του hazard. Βλέπουμε ότι όταν το rt_IF είναι ίσο με το rt_DEC, οι έξοδοι pc_enable και instreg_en είναι “0” ενώ η έξοδος cs_sel είναι “1”.

Συμπεράσματα

- Σε έναν pipelined επεξεργαστή προκύπτουν data hazards, control hazards και structural hazards τα οποία εμποδίζουν την σωστή εκτέλεση του προγράμματος και αντιμετωπίζονται με το forwarding και το stall.
- Σε ένα pipelined επεξεργαστή δεν χρειάζεται να τελειώσει η εκτέλεση μιας εντολής για να ξεκινήσει η εκτέλεση της επόμενης σε αντίθεση με έναν επεξεργαστή με datapath πολλαπλών κύκλων.
- Το κύκλωμα του control είναι συνδυαστικό και όχι FSM.