ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

Αναφορά μαθήματος ΗΡΥ302

Εργασία #2: Σχεδίαση επεξεργαστή πολλαπλών κύκλων και μετατροπή του σε pipeline

Γεώργιος Φραγγιάς 2018030086
gfrangias@isc.tuc.gr

Διδάσκων Καθηγητής:
Σωτήριος Ιωαννίδης

Υπεύθυνος Εργαστηρίου:
Κυπριανός Παπαδημητρίου

Πολυτεχνειο Κρητης

Τμημα Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων

Μάιος 2021

Σκοπός εργαστηριακής άσκησης

Ο σκοπός της συγκεκριμένης εργαστηριακής άσκησης είναι να κατανοηθούν, να σχεδιαστούν και να υλοποιηθούν επεξεργαστές πολλαπλών κύκλων και τύπου σωλήνα.

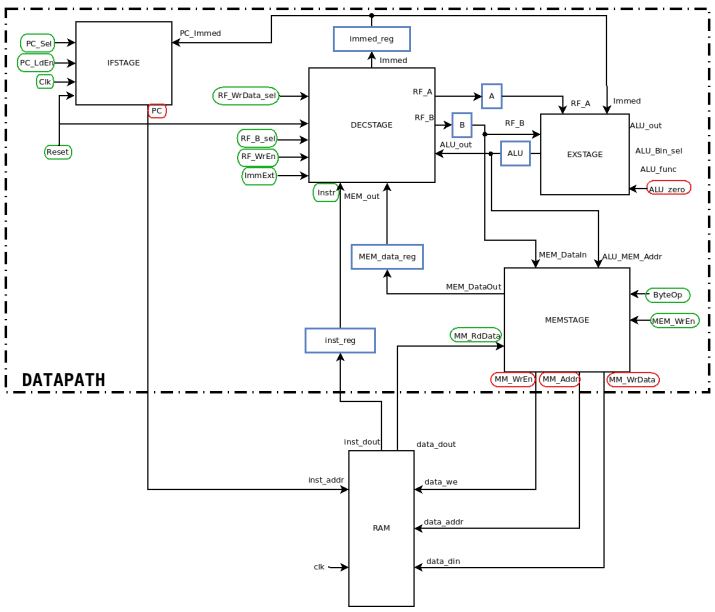
Προεργασία

Πριν από την υλοποίηση της εργασίας ήταν απαραίτητη η πολλή λεπτομερής κατανόηση της λειτουργίας των επεξεργαστών που ζητούνται. Ειδικότερα στον επεξεργαστή πολλαπλών κύκλων χρειάζεται ο σχεδιασμός μιας μηχανής πεπερασμένων καταστάσεων για τον καθορισμό των σταδίων τα οποία θα χρησιμοποιούνται από την κάθε εντολή. Στον επεξεργαστή τύπου σωλήνα ήταν αναγκαία η παρατήρηση όλων των επικίνδυνων συνδυασμών εντολών και η αντιμετώπισή τους με χρήση των τεχνικών προώθησης και καθυστέρησης.

Περιγραφή

4η Φάση της Άσκησης 2

Για την 4η φάση χρειάστηκε να προστεθούν στο `DATAPATH` καταχωρητές, ώστε να αποθηκεύονται οι τιμές των σημαντικών σημάτων μεταξύ σταδίων. Στο παρακάτω σχήμα φαίνονται στα μπλε πλαίσια.



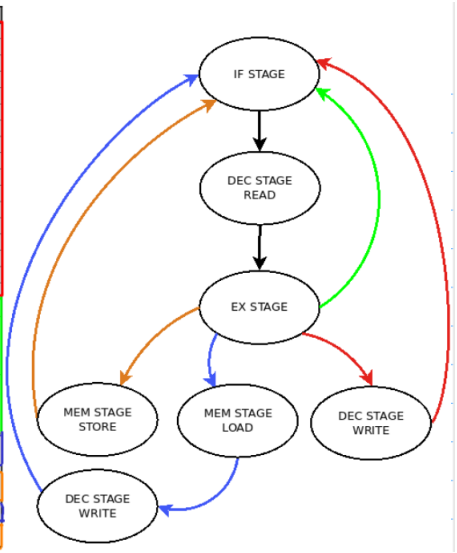
Έπειτα, ήταν απαραίτητα να σχεδιαστεί σχολαστικά η μηχανή πεπερασμένων καταστάσεων η οποία θα αποφάσιζει στο `CONTROL` ποια θα είναι τα στάδια που θα ακολουθηθούν ανάλογα με την εντολή που εκτελείται. Η μηχανή αυτή φαίνεται στο παρακάτω σχήμα. Τα μαύρα βέλη είναι διαδρομή η οποία ακολουθείται από όλες τις εντολές. Τα χρωματιστά βέλη είναι οι διαδρομές που ακολουθούν οι εντολές του πίνακα με χρωματική αντιστοίχηση.

Opcode	FUNC	ΕΝΤΟΛΗ	ΠΡΑΞΗ
100000	110000	add	RF[rd] ← RF[rs] + RF[rt]
100000	110001	sub	RF[rd] ← RF[rs] - RF[rt]
100000	110010	and	RF[rd] ← RF[rs] AND RF[rt]
100000	110011	or	RF[rd] ← RF[rs] | RF[rt]
100000	110100	not	RF[rd] ← ! RF[rs]
100000	110101	nand	RF[rd] ← RF[rs] NAND RF[rt]
100000	110110	nor	RF[rd] ← RF[rs] NOR RF[rt]
100000	111000	sra	RF[rd] ← RF[rs] >>1
100000	111001	srl	RF[rd] ← RF[rs] >>1 (Logical, zero fill MSB)
100000	111010	sll	RF[rd] ← RF[rs] <<1 (Logical, zero fill LSB)
100000	111100	rol	RF[rd] ← Rotate left(RF[rs])
100000	111101	ror	RF[rd] ← Rotate right(RF[rs])
111000	-	li	RF[rd] ← *SignExtend*(Imm)
111001	-	lui	RF[rd] ← Imm << 16 (zero-fill)
110000	-	addi	RF[rd] ← RF[rs] + *SignExtend*(Imm)
110010	-	nandi	RF[rd] ← RF[rs] NAND *ZeroFill*(Imm)
110011	-	ori	RF[rd] ← RF[rs] | *ZeroFill*(Imm)
111111	-	b	PC ← PC + 4 + (*SignExtend*(Imm) << 2)

000000	-	beq	if (RF[rs] == RF[rd])
			PC ← PC + 4 + (*SignExtend*(Imm) << 2)
			else
			PC ← PC + 4

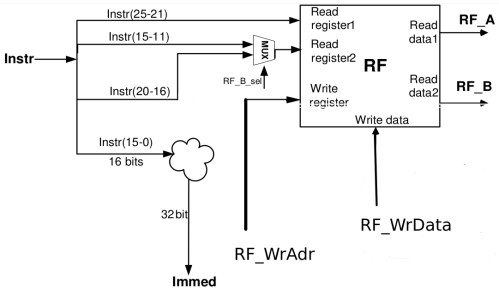
000001	-	bne	if (RF[rs] != RF[rd])
			PC ← PC + 4 + (*SignExtend*(Imm) << 2)
			else
			PC ← PC + 4

000011	-	lb	RF[rd] ← *ZeroFill*(31 downto 8) & MEM[RF[rs] + *SignExtend*(Imm)](7 downto 0)
000111	-	sb	MEM[RF[rs] + *SignExtend*(Imm)] ← *ZeroFill*(31 downto 8) & RF[rd] (7 downto 0)
001111	-	lw	RF[rd] ← MEM[RF[rs] + *SignExtend*(Imm)]
011111	-	sw	MEM[RF[rs] + *SignExtend*(Imm)] ← RF[rd]

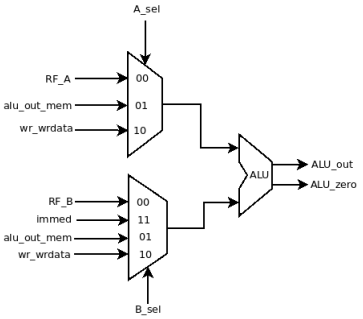


5η Φάση της Άσκησης 2

Για την 5η φάση χρειάστηκε σε πρώτη φάση να γίνουν κάποιες αλλαγές στα στάδια `DECSTAGE` και `EXSTAGE`.



[DEC]



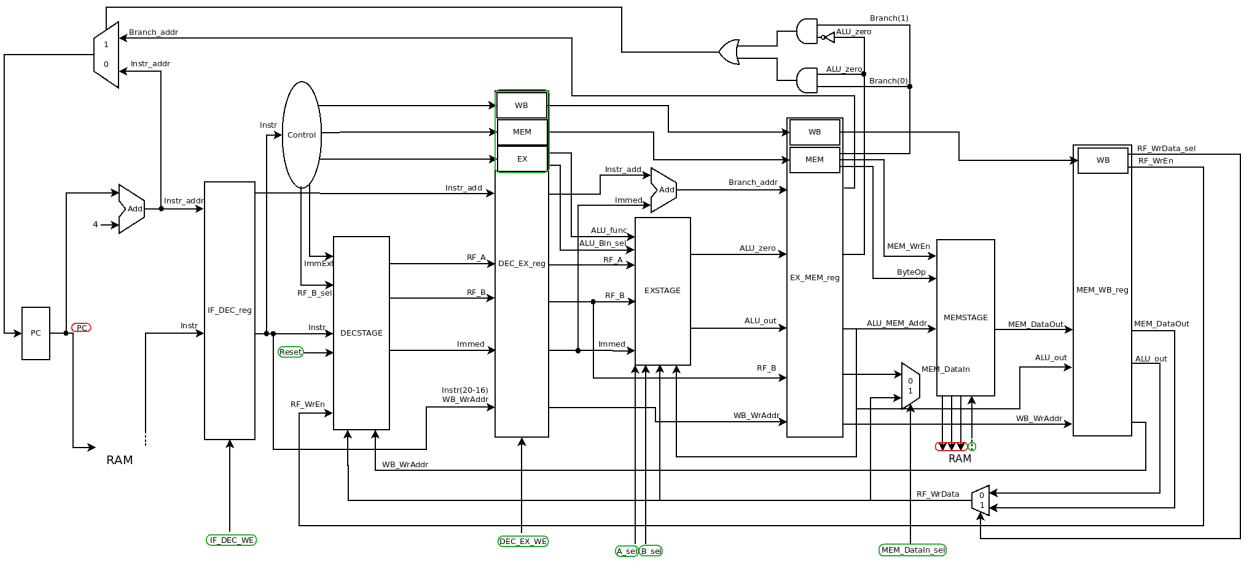
[EX]

Το `DECSTAGE` χρειάστηκε μία νέα είσοδο `RF_WrAddr` έτσι ώστε να παίρνει την διεύθυνση που χρειάζεται για writeback από την παλιά εντολή και όχι από την νέα που προσπαθεί να διαβάσει από το αρχείο καταχωρητών. Επίσης, ο πολυπλέκτης επιλογής εισόδου για εγγραφή μεταφέρθηκε στο `DATAPATH` για να είναι διαθέσιμος για forwarding.
Το `EXSTAGE` χρειάστηκε αλλαγές στους πολυπλέκτες, ώστε να υποστηρίζει forwarding. Για την υλοποίηση του forwarding χρησιμοποιούνται τα σήματα ελέγχου `A_sel, B_sel` τα οποία αποφασίζουν εάν θα γίνει forwarding της τιμής που πρόκειται να εγγραφεί από την προηγούμενη εντολή στο `EXSTAGE` της επόμενης.

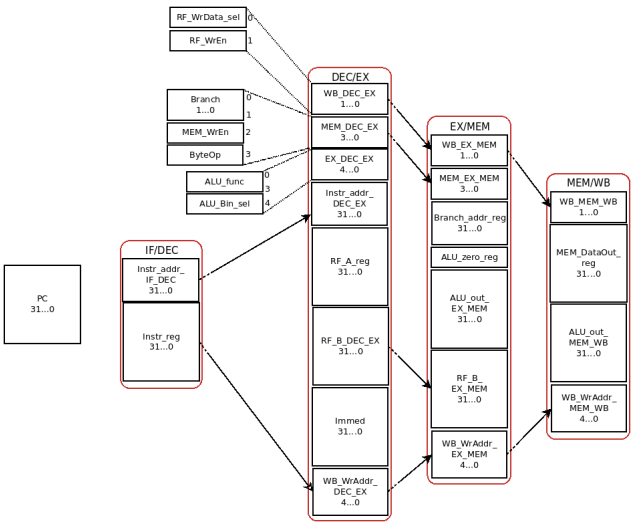
Απαραίτητες αλλαγές έγιναν και στο `DATAPATH` όπου όλα σχεδόν τα ενδιάμεσα σήματα αποθηκεύονται σε καταχωρητές μεταξύ των σταδίων. Επίσης χρειάστηκε να μεταφέρεται η διεύθυνση της επόμενης εντολής, ώστε να χρησιμοποιείται στην περίπτωση εντολής διακλάδωσης. Η περιπτώσεις αυτές ελέγχονται από το `2-bit` σήμα ελέγχου `Branch`

	Branch	
b	1	1
beq	0	1
bne	1	0
nothing	0	0

Το ολοκληρωμένο σχήμα του DATAPATH φαίνεται παρακάτω.



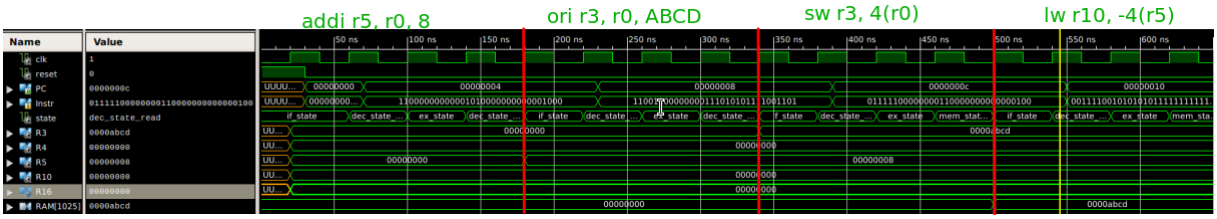
Οι μεγάλοι καταχωρητές(IF_DEC_reg, DEC_EX_reg) περιέχουν μικρότερους καταχωρητές και οι καταχωρητές WB, MEM, EX μεταφέρουν τα σήματα ελέγχου του CONTROL.
Στο παρακάτω σχήμα φαίνεται η σύνθεσή τους.



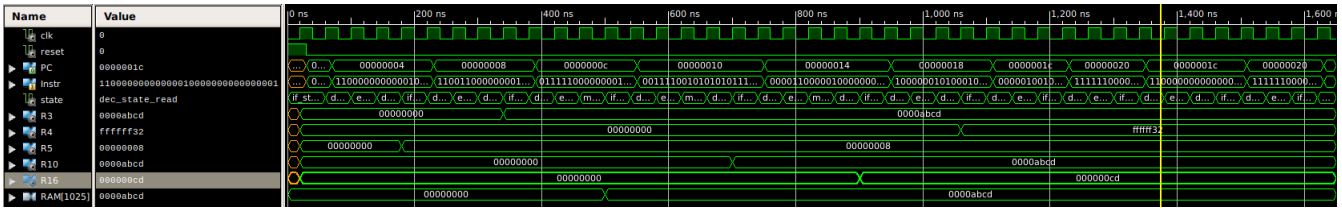
Κυματομορφές-Προσομοίωση

Δυστυχώς δεν πρόλαβα να δοκιμάσω τους επεξεργαστές σε σετ που περιέχει όλες τις εντολές, οι παρακάτω κυματομορφές εξετάζουν την λειτουργία των επεξεργαστών στο σετ εντολών που δώθηκε στην εργασία 1.

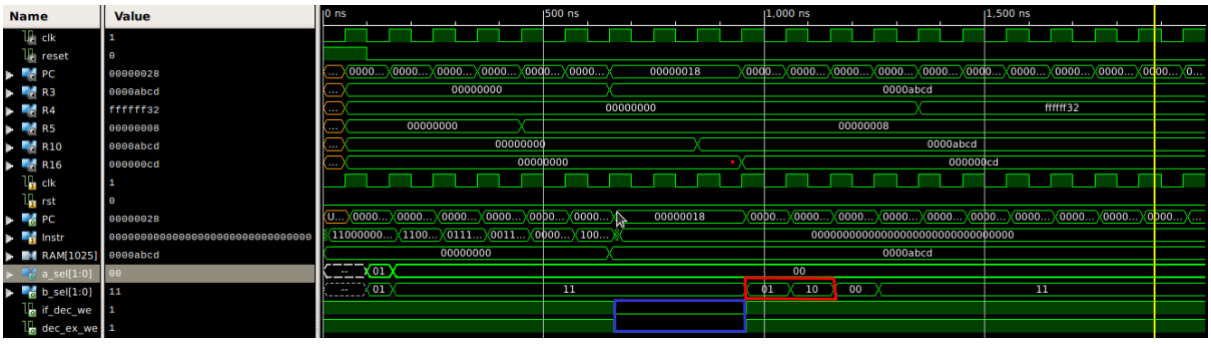
Στην παρακάτω κυματομορφή φαίνεται καθαρά το πως αλλάζουν τα στάδια με βάση την εντολή που εκτελείται στον επεργαστή πολλαπλών κύκλων.



Η παρακάτω κυματομορφή είναι ολόκληρο το testbench του επεξεργαστή πολλαπλών κύκλων.



Η επόμενη κυματομορφή είναι αυτή του επεργαστή τύπου σωλήνα. Σε αυτήν την κυματομορφή φαίνεται πως στην χρονική περίοδο του μπλε πλαισίου γίνεται stall. Το stall γίνεται για να προλάβει να εκτελεστεί η `lb r16, 4(r0)` και να γράψει στον καταχωρητή `r16` πριν εκτελεστεί η εντολή `nand r4, r10, r16` η οποία χρειάζεται να διαβάσει τον `r16`. Επίσης, στο κόκκινο πλαίσιο φαίνονται οι αλλαγές του `B_sel` λόγω της ανάγκης forwarding. Στην προκειμένη περίπτωση ο `r5` γράφεται στην εντολή `addi r5, r0, 8`. Και για αυτό τον λόγο θα πρέπει να γίνει forward της τιμής του στην εντολή `lw r10, -4(r5)`, όπου πρέπει να διαβαστεί το περιεχόμενό του. Δεν έχω προλάβει να υλοποιήσω το stall σε περιπτώσεις branch.



Συμπεράσματα

Μετά την υλοποίηση αυτής της άσκησης έγιναν καλά κατανοητοί οι επεξεργαστές πολλαπλών κύκλων και τύπου σωλήνα. Είναι πολύ σημαντικό το ότι παρατηρήθηκε το πως μπορεί ο κάθε τύπος επεξεργαστή να βελτιώσει την καθυστέρηση στον χρόνο. Παρατηρείται, λοιπόν, πως ο επεξεργαστής ενός κύκλου παρόλο που εκτελεί μία εντολή ανά κύκλο, ο κύκλος ρολογιού θα πρέπει να είναι αρκετά μεγάλος, ώστε να προλαβαίνει να εκτελεστεί η πιο χρονοβόρα εντολή. Στον επεξεργαστή πολλαπλών κύκλων ένας κύκλος ισοδυναμεί ένα στάδιο και άρα είναι πολύ

πιο σύντομος. Οι εντολές που χρησιμοποιούν λίγους κύκλους ρολογιού δεν χρειάζεται πλέον να περιμένουν άσκοπα και άρα ο επεξεργαστής είναι πολύ πιο γρήγορος. Ο επεξεργαστής τύπου σωλήνα χρησιμοποιεί την λειτουργικότητα του επεξεργαστή πολλαπλών κύκλων με την διαφορά ότι πολλές διαφορετικές εντολές μπορούν να εκτελεστούν ταυτόχρονα σε διαφορετικά στάδια. Αυτό, με εξαίρεση τις περιπτώσεις stall και την πρώτη εντολή, σημαίνει ουσιαστικά ότι κάθε εντολή βγάζει τελειώνει έναν μικρό κύκλο ρολογιού μετά την προηγούμενη.

Κώδικας

Αυτό είναι ένα παράδειγμα από την FSM του `CONTROL` του επεξεργαστή πολλαπλών κύκλων. Όταν βρίσκεται στο `DECSTAGE` αλλάζουν οι τιμές των παρακάτω καταχωρητών ώστε να ταιριάζουν στο `EXSTAGE` και προχωράει η FSM στο `EXSTAGE`.

```
when dec_state_read =>

        PC_LdEn_con <= '0';
        RF_WrEn_con <= '0';
        MEM_WrEn_con <= '0';
        instr_reg_we_con <= '0';
        immed_reg_we_con <= '0';
    A_reg_we_con <= '0';
        B_reg_we_con <= '0';
        ALU_reg_we_con <= '1';
        MEM_data_reg_we_con <= '0';

        State <= ex_state;
```