26/3/2017

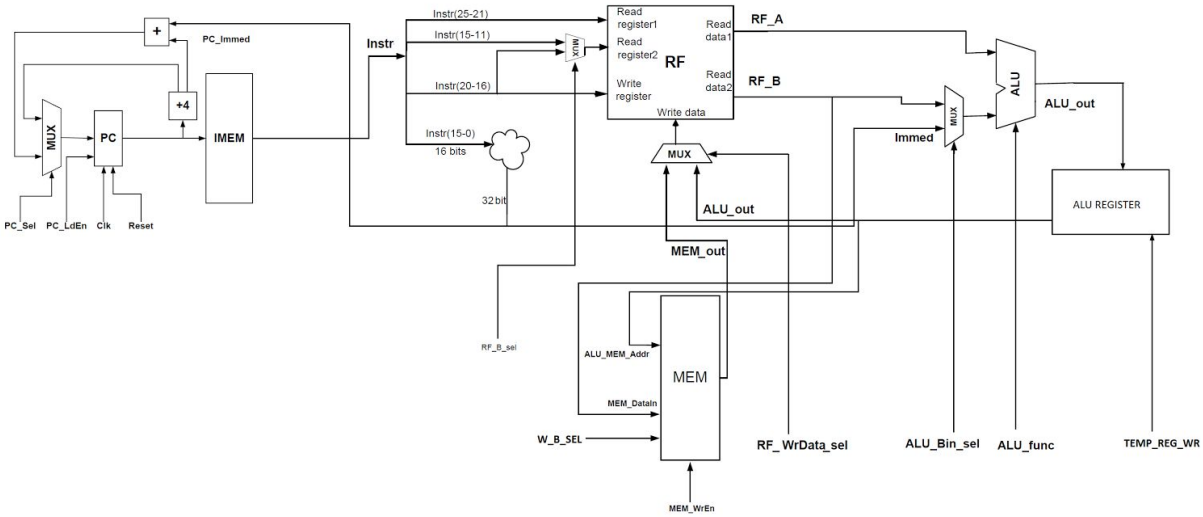
Αναφορά Εργαστηρίου 3

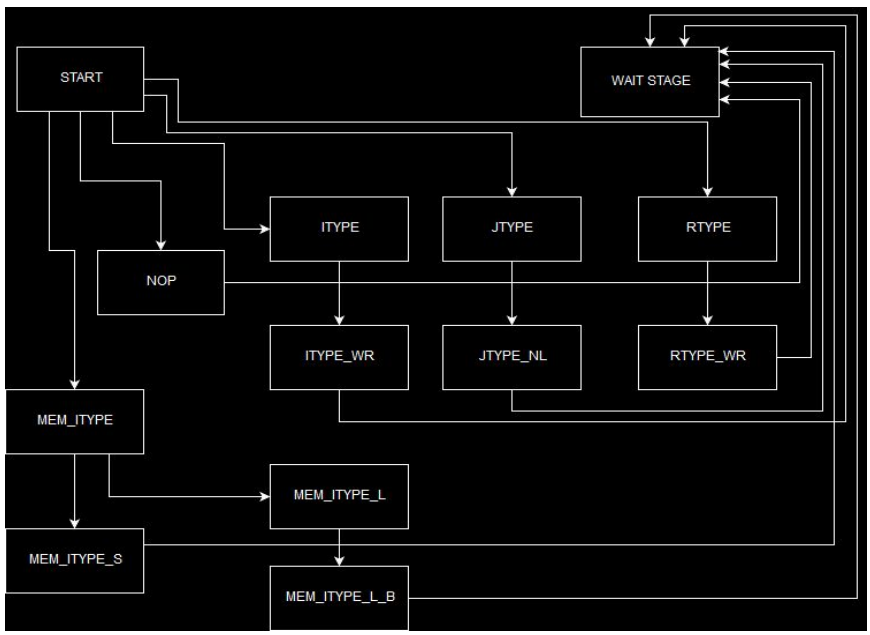
Κωδικός Ομάδας **LAB31231465**

Διαλεκτάκης Γιώργος
Βαϊλάκης Απόστολος Νικόλαος

Προεργασία

Ως προεργασία του 3ου εργαστηρίου μας ζητήθηκε ένα σχηματικό διάγραμμα του ολοκληρωμένου **Datapath** όπου φαίνονται οι συνδέσεις μεταξύ των βαθμίδων που υλοποιήσαμε σε προηγούμενα εργαστήρια καθώς και της μηχανής πεπερασμένων καταστάσεων του **Control.**





Περιγραφή Άσκησης

Σκοπός της 3ης εργαστηριακής άσκησης ήταν η σχεδίαση της μονάδας Ελέγχου(**Control**) του **Datapath** το οποίο ολοκληρώσαμε σχεδιάζοντας τις συνδέσεις των βαθμίδων που είχαμε υλοποιήσει στα πλαίσια του 2ου εργαστηρίου. Τελικός σκοπός η κατασκευή ενός Επεξεργαστή(**Processor**) πολλαπλών κύκλων.

Datapath:

Όσον αφορά το datapath, τα πράγματα είναι "απλά". Αποτελείται από το **IFSTAGE**, **DECSTAGE, ALUSTAGE** και **MEMSTAGE**(λεπτομέρειες στην αναφορά του 2ου εργαστηρίου). Όλα τα παραπάνω ενώθηκαν για να αποτελέσουν το datapath. Σε αυτό επίσης προστέθηκε ένας καταχωρητής στην έξοδο της ALU με σκοπό την σταθεροποίηση της εξόδου της και την αποφυγή προβλημάτων feedback. Ακόμα, προστέθηκε ένα σήμα W_B_SEL στην MEMSTAGE για την επιλογή ανάμεσα σε lb, sb ή lw, sw.

Control:

Η μηχανή πεπερασμένων καταστάσεων που υλοποιεί το Control του Datapath αποτελείται από τις εξής καταστάσεις:

START: Εδώ ξεκινάει η λειτουργία της FSM όταν έρχεται μία νέα εντολή.

NOP: Οδηγούμαστε εδώ όταν η εντολή είναι "00000000000000000000000000000000" και το Control δεν ενεργοποιεί κανένα σήμα αλλά διαβάζει την επόμενη εντολή.

WAIT_STAGE: Δημιουργήθηκε για την αφιέρωση ενός επιπλέον κύκλου για τη σωστή μετάβαση από την μία εντολή στην επόμενη.

RTYPE: Εδώ οδηγούνται οι εντολές με Opcode "100000"(add,sub,and,not,or,sra,sll,srl,rol,ror)
RTYPE_WR: Αφορά τον κύκλο στον οποίο γράφουμε στην Register File και συγκεκριμένα στον καταχωρητή rd το αποτέλεσμα της RTYPE εντολής.

ITYPE: Εδώ οδηγούνται οι εντολές με Opcode "111000", "111001", "110000", "110010", "110011"(li, lui ,addi, andi, ori αντίστοιχα).
ITYPE_WR: Αφορά και εδώ τον κύκλο στον οποίο γράφουμε στην Register File και συγκεκριμένα στον καταχωρητή rd το αποτέλεσμα της εντολής με Immediate.

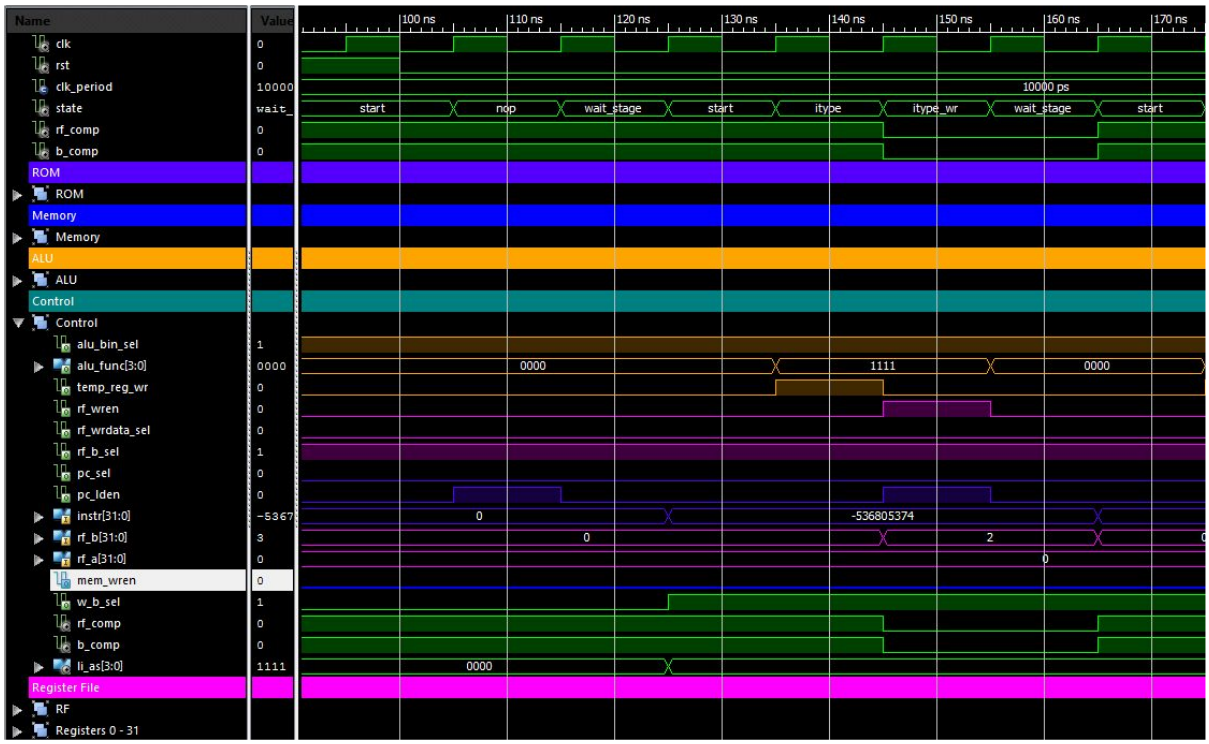
JTYPE: Εδώ φτάνουν οι εντολές που εκτελούν διακλάδωση και έχουν Opcode "111111", "000000", "000001"(b,beq,bne).
JTYPE_NL: Η κατάσταση στην οποία αν επαληθεύεται η συνθήκη της εντολής διακλάδωσης, τότε το πρόγραμμα διακλαδώνεται και ο PC αυξάνεται κατάλληλα έτσι ώστε να διαβάσουμε την σωστή εντολή και όχι την ακριβώς επόμενη. Η αύξηση του PC επιλέγεται από το σήμα PC_SEL το οποίο υλοποιείται με συνδυαστική λογική βασιζόμενη στον Opcode

και στις εξόδους RF_A και RF_B.
MEM_ITYPE: Εδώ οδηγούνται εντολές που αφορούν ανάγνωση και εγγραφή στη μνήμη με Opcode “000011”, “000111”, “001111”, “011111”.
MEM_ITYPE_L: Κατάσταση για τις εντολές ανάγνωσης(lb, lw).
MEM_ITYPE_L_B: Ενδιάμεση κατάσταση για να προλαβαίνει η μνήμη να βγάζει τα δεδομένα στην έξοδο.
MEM_ITYPE_S: Κατάσταση για τις εντολές εγγραφής(sb, sw).

Η FSM βγάζει ως έξοδο τα σήματα ALU_Bin_Sel, ALU_Func, RF_WrEn, RF_WrData_Sel, RF_B_Sel, Temp_Reg_Wr, PC_SEL, PC_LdEn, Mem_WrEn και W_B_Sel και τα ενεργοποιεί ανάλογα με την κατάσταση στην οποία βρίσκεται.

Κυμματομορφές:

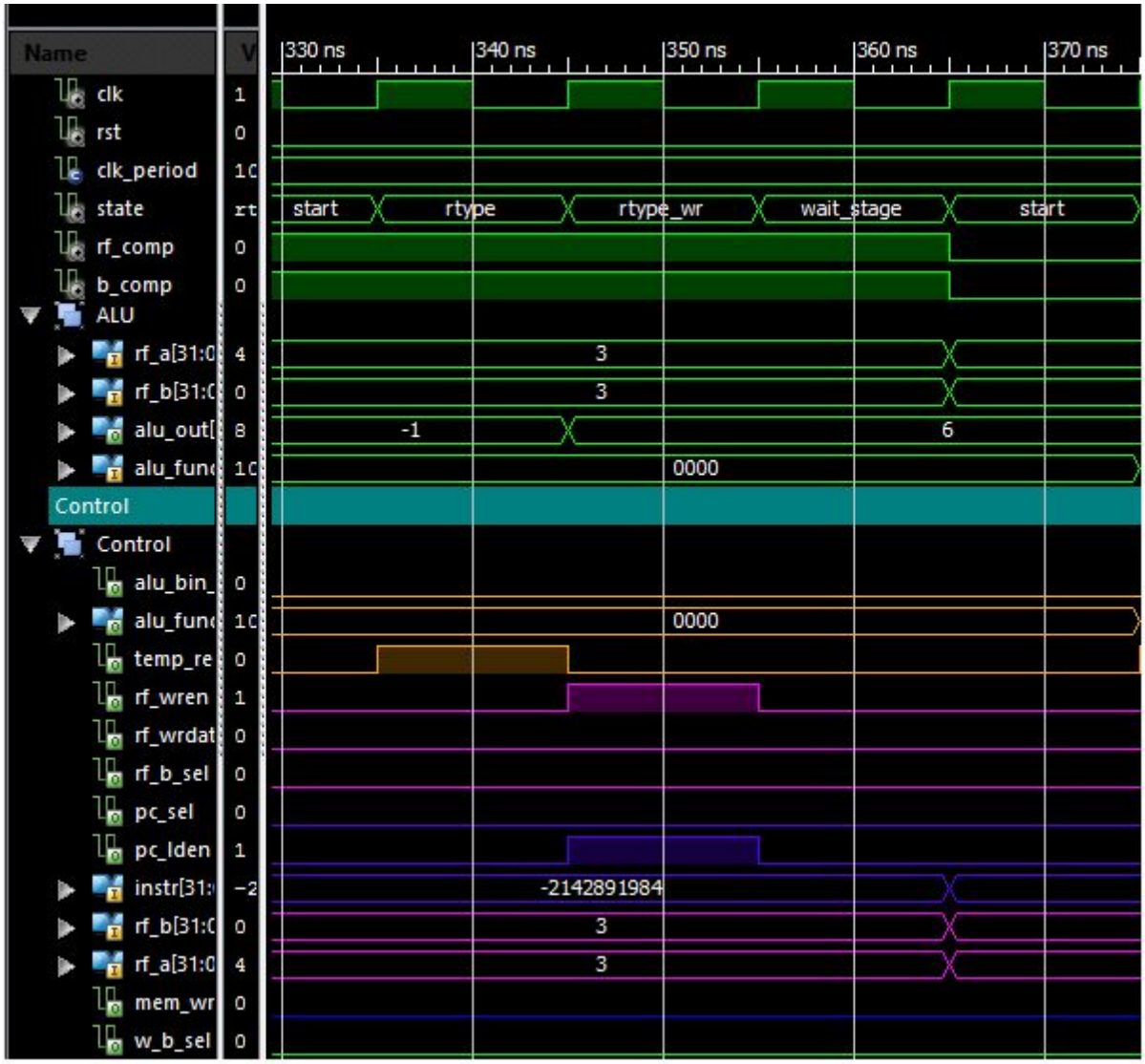
NOP , ITYPE



Παραπάνω βλέπουμε μία περίπτωση NOP. Σε αυτήν την περίπτωση βλέπουμε όλα τα σήματα του CONTROL στο 0 . Όλα πέραν του pc_lden για να καλέσουμε την επόμενη εντολή. αμέσως επόμενη είναι μια εντολή i_type και συγκεκριμένα η “li $1, 2 “. Σε αυτήν την εντολή μπορούμε να δούμε τα

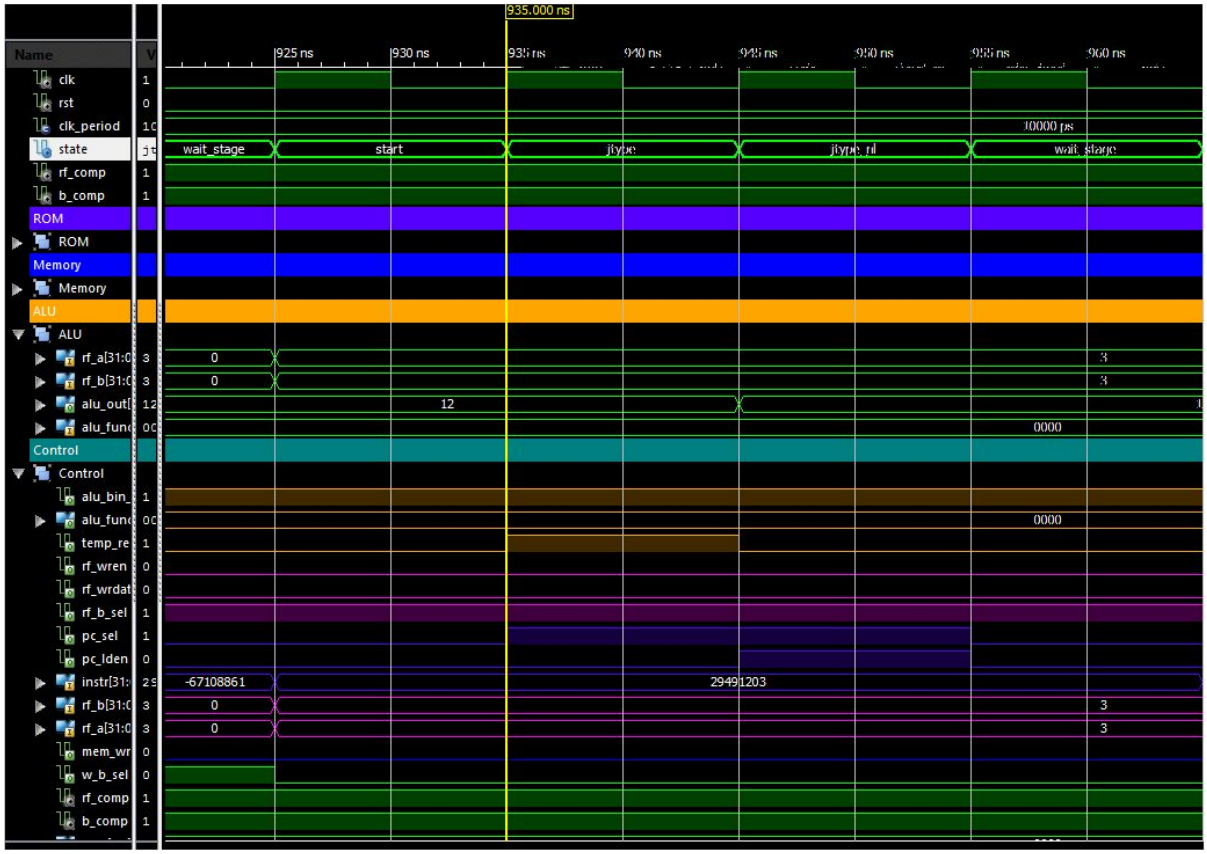
σήματα ελέγχου rf_wrdata_sel = 0 και rf_b_sel = 1 αφου έχουμε να κάνουμε με εντολή τύπου immediate. Ακόμη βλέπουμε οτι σήμα temp_reg_wr ενεργοποιείται δίνοντας μας στον επόμενο κύκλο τη έξοδο της η ALU. Τέλος στο state itype_wr ενεργοποιούνται ταυτόχρονα τα σήματα rf_wren και pc_lden για να εγγραφούν τα κατάλληλα δεδομένα στην register file και να αλλάξει το Instruction μετα απο 2 κύκλους. Ο χρόνος για αυτη την αλλαγή δίνεται απο την wait_stage.

RTYPE



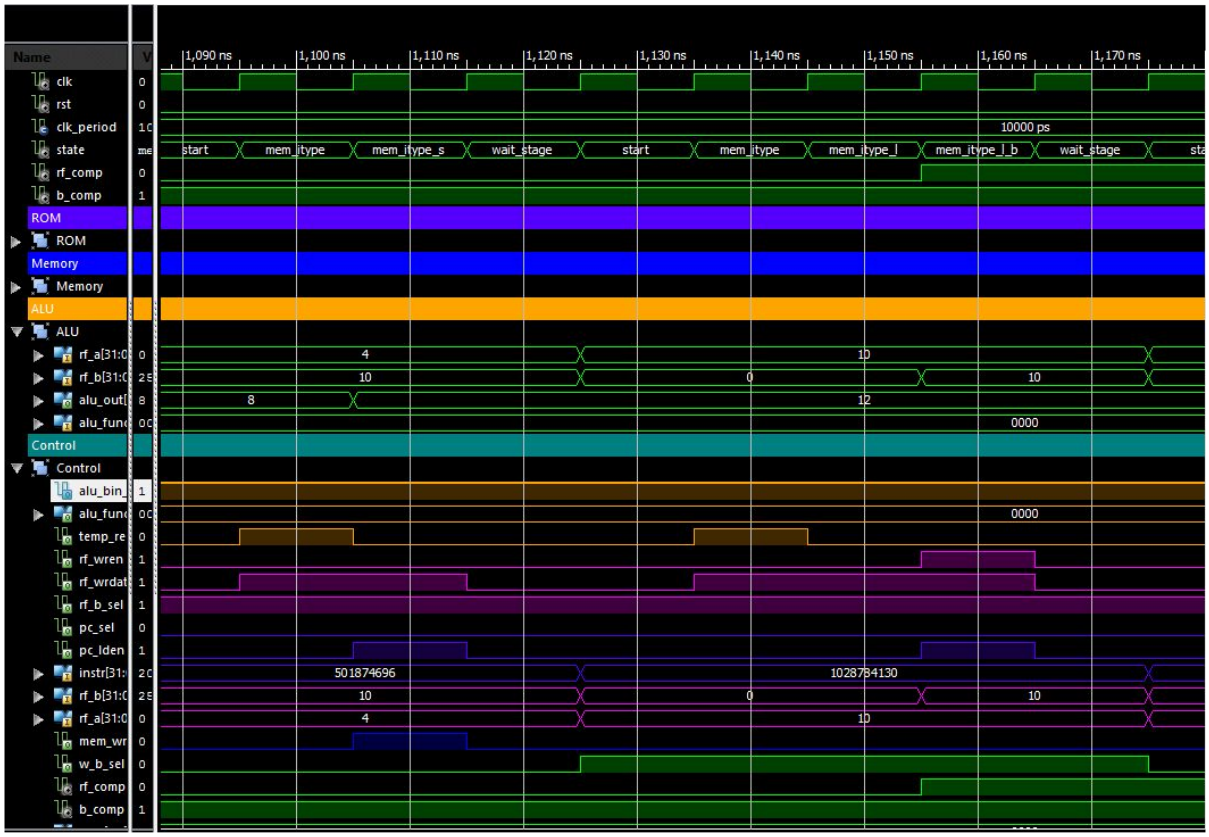
Παραπάνω βλέπουμε μια εντολή RTYPE και συγκεκριμένα την εντολή ADD. Μπορούμε να δούμε το function της ALU (0000 = ADD) . Παλι το σήμα temp_reg_wr ενεργοποιείται για την εξοδο των δεδομένων απο την ALU και όντας εντολή Rtype τα σηματα ελέγχου rf_wrdata_sel και rf_b_sel ειναι στο 0. Παρόμοια με την προηγούμενη αντολη, στην rtype_wr state ενεργοποιείται το rf_wren και το pc_lden για την καταγραφή των καινούργιων δεδομένων στην RF και την μεταβαση σε επόμενη εντολη.

JTYPE



Επόμενη εντολή προς ανάλυση είναι μια εντολή τύπου branch. Σε αυτήν την εντολή ενεργοποιείται η επιλογή pc_lden οπως σε καθε εντολή, όμως ενα σήμα pc_sel επιλέγει αν θα το PC θα αυξηθει κατα 4 ή κατα 4 + Immediate. Αυτο το σήμα υλοποιείται με απλή λογική χρησιμοποιώντας εναν comparator αναμεσα σε RF_A και RF_B και τον opcode.

Mem_IType.



Σε αυτού του είδους τις εντολές αρχικά επιλέγεται απο την ALU η πράξη της προσθεσης. Ακόμη επιλέγεται ως δευτερη είσοδός της ο IMMEDIATE. Ακόμα οι εντολές Mem_Itype χωρίζονται σε Save και Load τύπου. Οι τύπου Load χρησιμοποιούν έναν ακόμη παραπάνω κύκλο, ο οποιος χρειαζεται απο την memory για να εξαγει τα δεδομένα της. Ανάλογα με την εντολή (save ή load τύπου) ενεργοποιείται το σήμα rf_wren ή mem_wren. Τέλος ένα σήμα W_B_Sel επιλέγει αν η μνήμη θα εισάγει και εξάγει ολα τα byte τον δεδομένων ή αν θα μηδενίζει τα 3 most significant byte. Αυτό το σήμα εξάγεται απο την Control βασιζόμενο στο opcode.

Συμπεράσματα:

Φθάνοντας πλέον στα μισά των συνολικών εργαστηρίων του μαθήματος, υλοποιήσαμε έναν απλό CHARIS επεξεργαστή πολλαπλών κύκλων και επιβεβαιώσαμε την σωστή λειτουργία του με ένα πρόγραμμα που μας δόθηκε έτοιμο.