ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ
ΗΡΥ 312

ΕΡΓΑΣΤΗΡΙΟ 2
Αναφορά Εργαστηριακής Άσκησης

Ομάδα Εργασίας **LAB31220188**

α/α	Α.Μ.	Ονοματεπώνυμο
1	2011030010	Χριστοδουλου Θεοφιλος
2	2011030009	ΚΑΡΙΜΠΙΔΗΣ ΔΙΟΝΥΣΗΣ

Το 2ο είχε σκοπό τη δημιουργία των components ενός datapath ικανού να εκτελέσει τις εντολές του CHARIS που δίνονται. Οι αντίστοιχοι κώδικες βρίσκονται στα συννημένα αρχεία.
Παρατηρώντας το op code της κωδικοποίησης, εύκολα καταλαβαίνουμε ότι οι R-type εντολές κωδικοποιούνται με οπ κόουντ 100000 και η ακριβής λειτουργία της ALU καθορίζεται από τα 4 μεσαία bits του func.

IFSTAGE

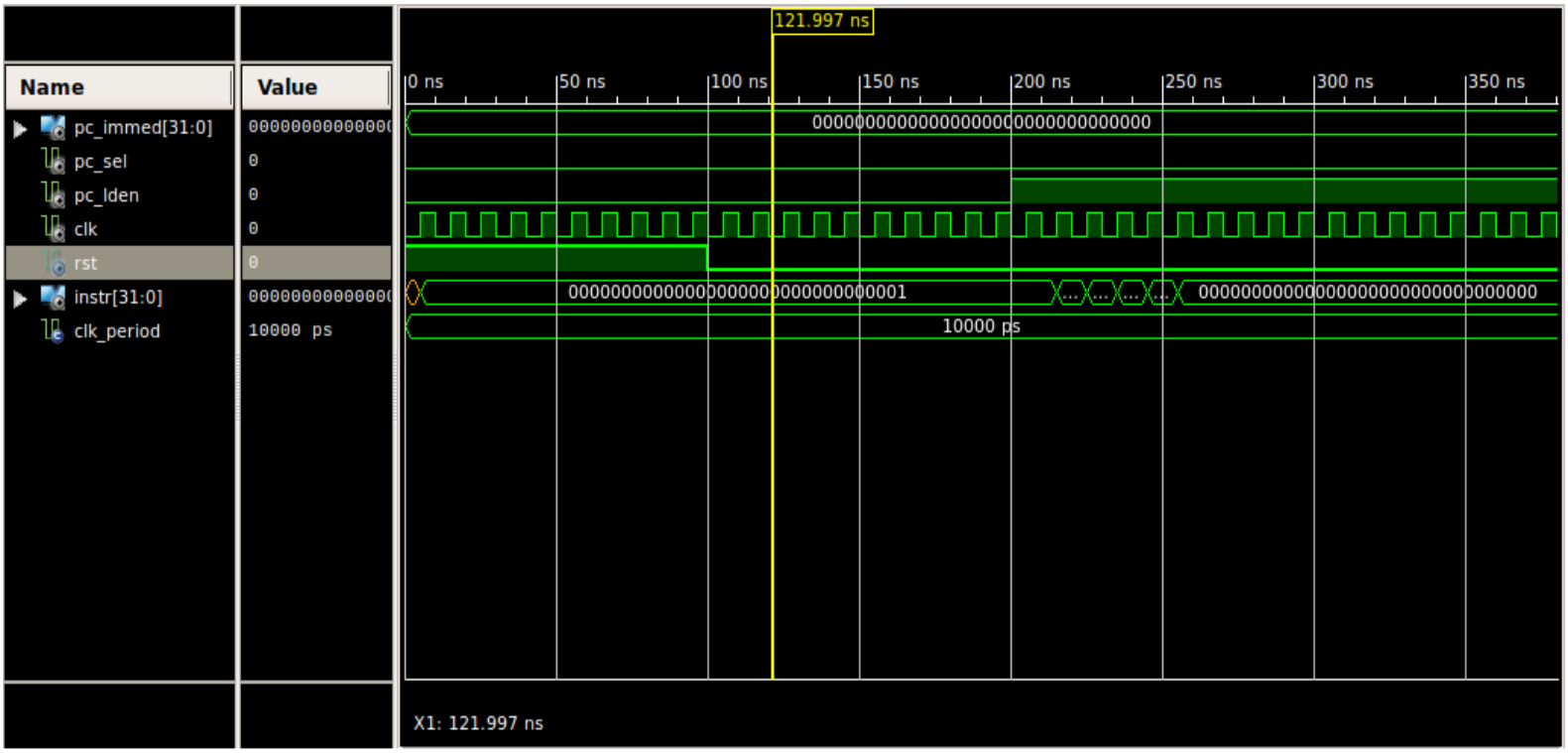
Το component IFSTAGE εκτελεί -στην ουσία- τη διαδικασία ανάκλησης εντολών. Το διάβασμά τους γίνεται από μία **μνήμη ROM** 1024 θέσεων των 32 bits, στην οποία είναι αποθηκευμένες οι εντολές, η αρχικοποίηση των οποίων επιτυγχάνεται με το συνοδευτικό αρχείο rom.data.

Ένας καταχωρητής, ο **Programm Counter** χρησιμοποιείται για να υποδεικνύουμε στη μνήμη τη θέση στην οποία βρίσκεται η εντολή που θέλουμε να εκτελέσουμε.

Από κει και πέρα, χρησιμοποιούμε έναν **αυξητή (incrementor)** που αυξάνει σε κάθε κύκλο τον PC κατά 4 και υλοποιεί το πέρασμα στην αμέσως επόμενη εντολή,και έναν **αθροιστή**. Ο αθροιστής προσθέτει το αποτέλεσμα του incrementor με το Immediate(32 bit- Sign extended -Shifted left 2), παράγοντας το σωστό αποτέλεσμα για branch εντολές (μετάβαση Immediate εντολές κάτω).

Ένας **πολυπλέκτης**, τέλος, χρησιμεύει στο να επιλέξουμε κατά πόσο θα αυξήσουμε τον PC, κατά 4 (sel=0) ήήήήή κατά 4+Immed*4(sel=1).
Ως έξοδος του component περνάει η 32 bit εντολή που ανακλήθηκε.

Ακολουθεί μια δοκιμαστική λειτουργία της βαθμίδας ανάκλησης εντολών, κατά την οποία φαίνεται (μετά το reset), να περνά στην έξοδο -με κάθε χτύπο του ρολογιού- διαδοχικά τις εντολές που βρίσκονται στο αρχείο rom.data:



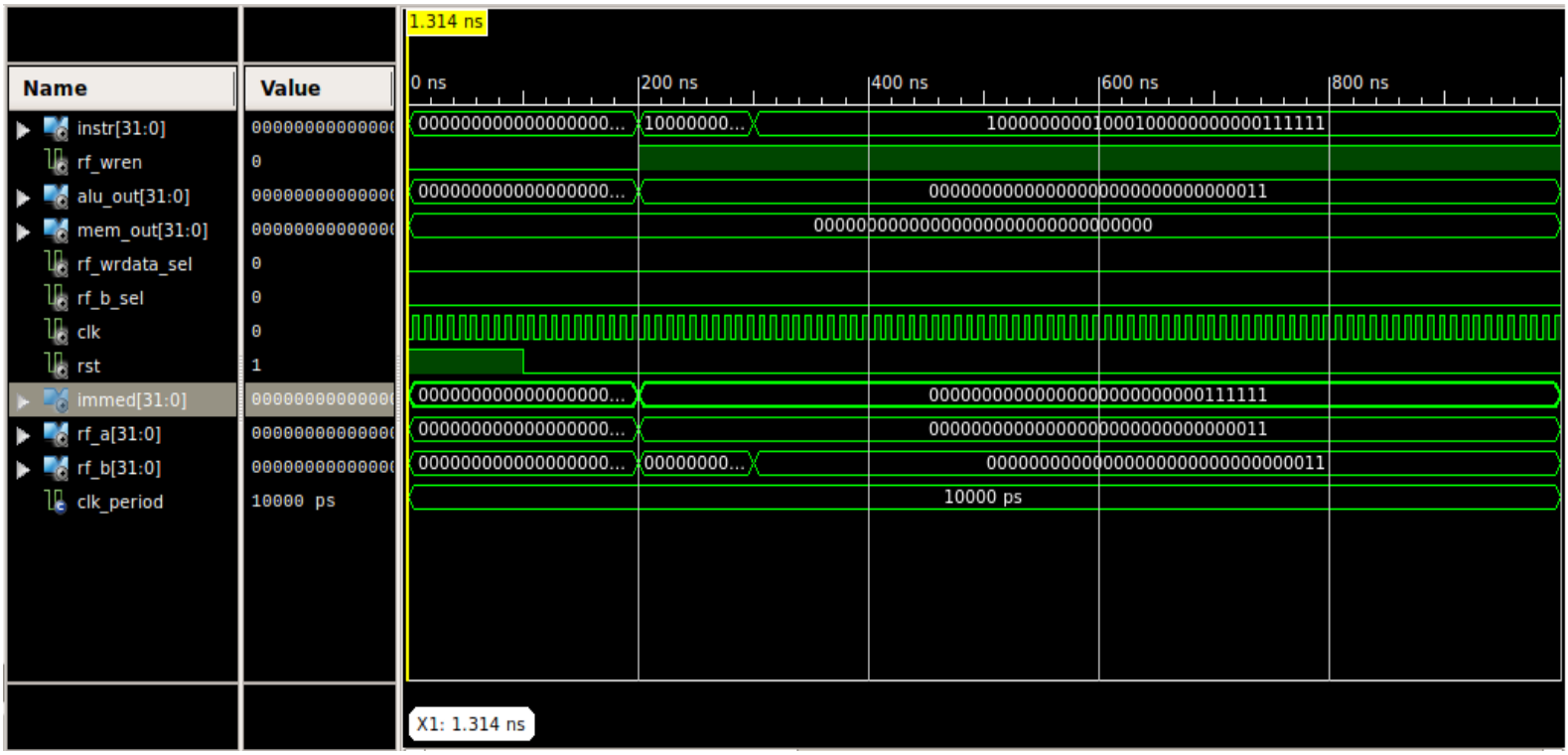
DECSTAGE

Η βαθμίδα αυτή περιέχει την **RegisterFile** που δημιουργήσαμε στο 1ο εργαστήριο, 2 πολυπλέκτες και έναν extender.

Η RF στην είσοδο ReadRegister1 παίρνει τον καταχωρητή Rs, στην είσοδο ReadRegister2 παίρνει το αποτέλεσμα του **πολυπλέκτη** ο οποίος επιλέγει την δεύτερη διεύθυνση διαβάσματος ανάλογα το format της εντολής. Έτσι για τις R-format εντολές η έξοδος του πολυπλέκτη αυτού είναι το Rt ενώ για τις I-format είναι ο Rd. Τα δεδομένα που γράφονται στην RF ¨μπαίνουν¨ στην είσοδο Write Data και θα είναι είτε η έξοδος της ALU είτε η έξοδος της MEM. Για την επιλογή αυτή υπάρχει ένας δεύτερος **πολυπλέκτης** που επιλέγει κάθε φορά την ανάλογη είσοδο δεδομένων στην RF.

Τέλος σε αυτό το στάδιο υλοποιήσαμε και τον **extender** που παίρνει ως είσοδο το Immed της εντολής αλλά και το Opcode. Έτσι η έξοδος ανάλογα το opcode θα είναι είτε το SignExtentionImmed είτε το ZeroFill Immed είτε το SignExtentionImmed, μετατοπισμένο κατά 2 θέσεις αριστερά για τις περιπτώσεις των brands. Ο τρόπος με τον οποίο κάνουμε το zeroExtention, signExtention ή μετατόπιση στο signExtention έχει περιγραφεί κατά την διάρκεια του εργαστηρίου και υπάρχει και στο αντίστοιχο vhd αρχείο στην αναφορά μας.

Ακολουθεί η δοκιμαστική του λειτουργία, κατά την οποία το decstage καλείται να αποκωδικοποιήσει μια εντολή που διάβασε και να χειριστεί σωστά τους κατάλληλους καταχωρητές:



ALUSTAGE

Η βαθμίδα αυτή περιέχει την **ALU** που δημιουργήσαμε στο 1ο εργαστήριο και έναν πολυπλέκτη. Η λειτουργία της ALU είναι γνωστή ενώ ο **πολυπλέκτης** χρησιμοποιείται ώστε να διαλέγουμε εάν στην 2η είσοδο της ALU θα εισάγουμε την έξοδο του 2ου καταχωρητή από την RF ή το Immed αφότου έχουμε κάνει το extention.

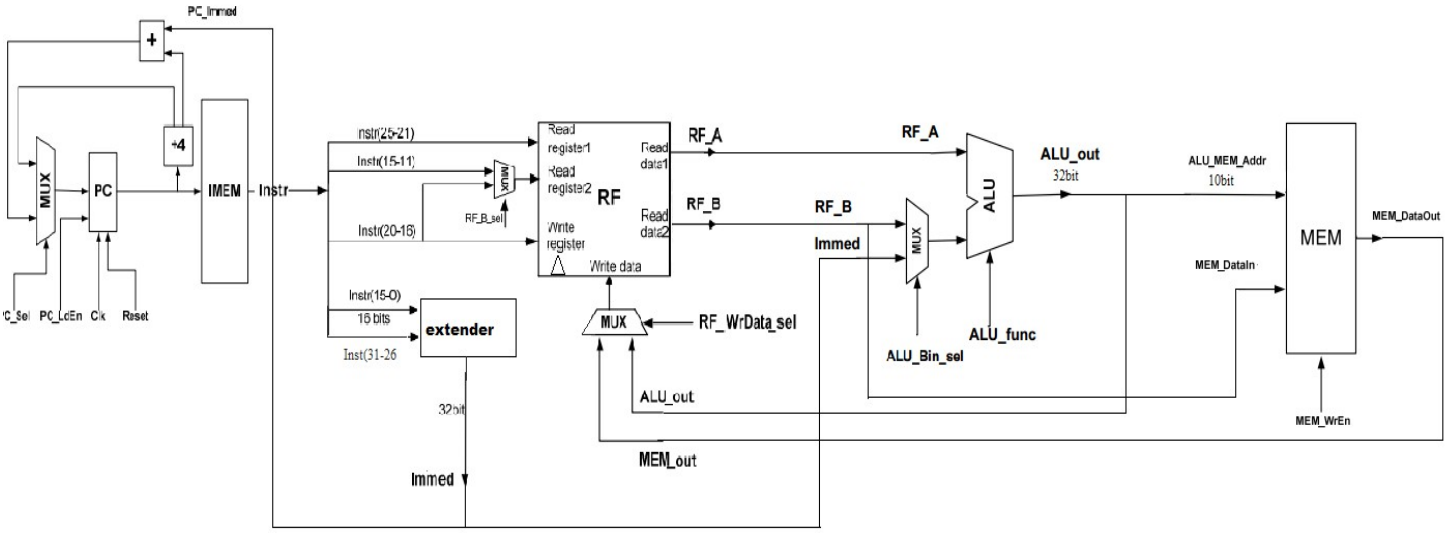
Το σήμα ελέγχου ALU_func είναι αυτό που επιλέγει την πράξη που θα γίνει ανάμεσα στα δεδομένα των 2 εισόδων και στην περίπτωσή μας το θεωρούμε ως είσοδο στο ALUSTAGE.

Μιας και τον έλεγχο της ALU τον έχουμε κάνει ήδη και στο προηγούμενο εργαστήριο και σε αυτό το επίπεδο χρησιμοποιείται χωρίς ιδιαίτερες αλλαγές δεν παραθέτουμε testBench για αυτό το επίπεδο.

Memstage

Στο τελευταίο component υλοποιήσαμε μια **RAM μνήμη** 1024 θέσεων των 32 bits, η οποία χρησιμοποιείται στις εντολές Load και Store, οι οποίες ανακαλούν ή αποθηκεύουν δεδομένα -αντίστοιχα- στη (data) μνήμη. Για την υλοποίησή της χρησιμοποιήθηκε ο κώδικας που δίνεται, με τη διαφορά ότι για τη διευθυνσιοδότηση της μνήμης χρησιμοποιήσαμε μόνο τα 10 LSB's από τα 32 του αποτελέσματος της ALU.

Παρακάτω φαίνεται ο συνολικός σχεδιασμός του datapath που πρωταγωνίστησε στην υλοποίηση αυτού του εργαστηρίου:



Χριστοδούλου Θεόφιλος **A.M.: 2011030010**

Καριμπίδης Διονύσης **A.M.: 2011030009**