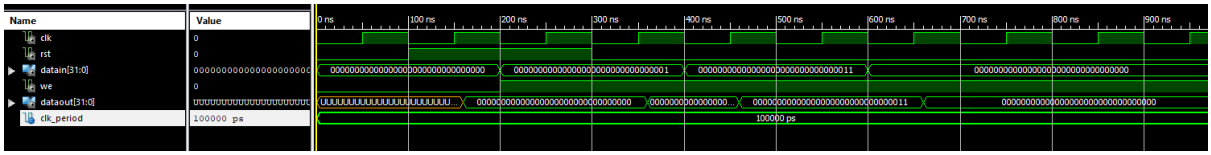
ΠΟΛΥΤΕΧΝΕΙΟ
ΚΡΗΤΗΣ

ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ 2022
ΑΝΑΦΟΡΑ ΠΡΩΤΗΣ ΕΡΓΑΣΙΑΣ

Αλέξανδρος Τερζής AM: **2013030184**

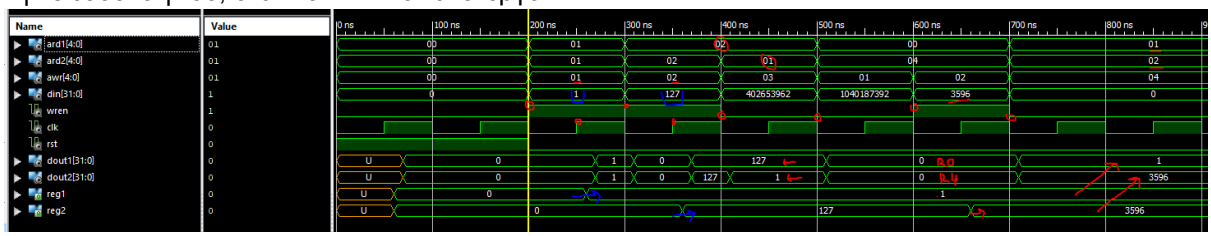
ΠΡΩΤΗ ΦΑΣΗ

Register: Είναι ένας καταχωρητής μήκους 32 μπιτ, που παίρνει μία τιμή ως είσοδο σε κάθε κύκλο ρολογιού και την αποθηκεύει όταν το WE είναι ενεργό. Την ίδια στιγμή στην έξοδο εμφανίζεται η τιμή που είναι αποθηκευμένη στον καταχωρητή.



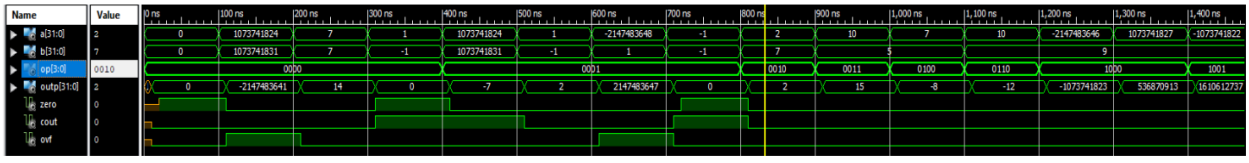
Στο Testbench εισάγω τιμές και αυτές εμφανίζονται στην έξοδο όταν έχω Enable.

Register File: Αποτελείται από 32 Register των 32 μπιτ. Μπορεί σε κάθε κύκλο ρολογιού να εμφανίζει τις τιμές 2 καταχωρητών που επιλέγουμε διαλέγοντας την διεύθυνσή τους, καθώς και να αποθηκεύει μια τιμή που εμείς εισάγουμε σε κάποιον από τους καταχωρητές 1 έως 31 διαλέγοντας την διεύθυνσή του, όταν το WrEn είναι ενεργό.



Μέσω των διευθύνσεων διαλέγω ποιους Register θέλω να εμφανίσω. Αν wren ενεργό γράφω στην διεύθυνση Awr των Register την τιμή Din.

ALU: Δέχεται ως είσοδο δύο καταχωρητές 32 μπιτ και εκτελεί αριθμητικές και λογικές πράξεις που εμείς επιλέγουμε εμφανίζοντας το αποτέλεσμα στην έξοδο. Η έξοδος Zero ενεργοποιείται όταν το αποτέλεσμα της πράξης είναι μηδέν. Οι έξοδοι Cout και Ovf μπορούν να ενεργοποιηθούν μόνο όταν γίνεται η πράξη της πρόσθεσης ή της αφαίρεσης. Η έξοδος Cout ενεργοποιείται όταν το αποτέλεσμα της πράξης δίνει κρατούμενο, και η Ovf όταν έχουμε υπερχείλιση.



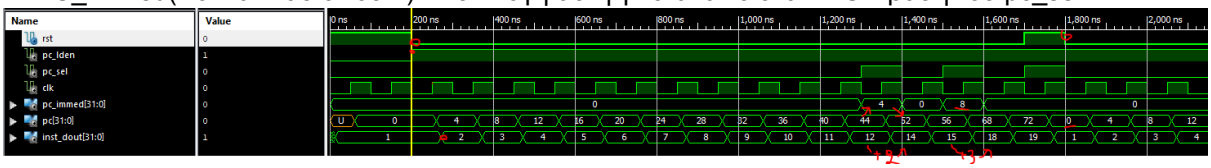
Στην προσομοίωση βλέπουμε αναλυτικά την πράξη της πρόσθεσης και της αφαίρεσης καθώς και τις οριακές τους περιπτώσεις. Οι υπόλοιπες πράξεις πιο αναλυτικά φαίνονται σε εικόνες στον φάκελο WAVEFORMS.

MUX_2to1: Είναι ένας generic mux που δέχεται ως εισόδους δύο σήματα N-μπιτ και βγάζει στην έξοδο ένα από τα δύο βάση επιλογής. Το N μπορούμε να το προσαρμόσουμε, default είναι 32.

DECODER_5to32: Είναι ο αποκωδικοποιητής που χρησιμοποιείται στην Register File ώστε να παίρνει το σήμα 5 μπιτ της διεύθυνσης και να το κάνει σε 32 μπιτ.

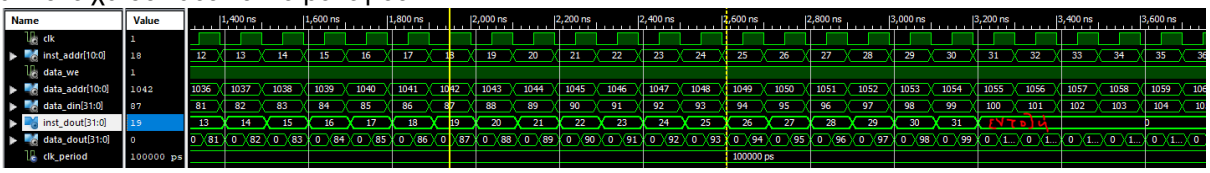
ΔΕΥΤΕΡΗ ΦΑΣΗ

IF_STAGE: Είναι η βαθμίδα ανάκλησης εντολών, δημιουργούμε έναν register(PC) ο οποίος μας δείχνει σε ποια διεύθυνση της μνήμης βρίσκεται η εντολή που πρέπει να κάνει ο επεξεργαστής. Ο PC αυξάνεται σε κάθε κύκλο ρολογιού κατά 4 γιατί οι εντολές είναι αποθηκευμένες στην διεύθυνση μνήμης σε θέσεις πολλαπλάσια του 4. Αν έχουμε εντολή τύπου branch ο PC αυξάνεται κατά 4+PC_Immed(πολλαπλάσιο του 4). Η επιλογή αυτή γίνεται από εναν MUX βάση του pc_sel.

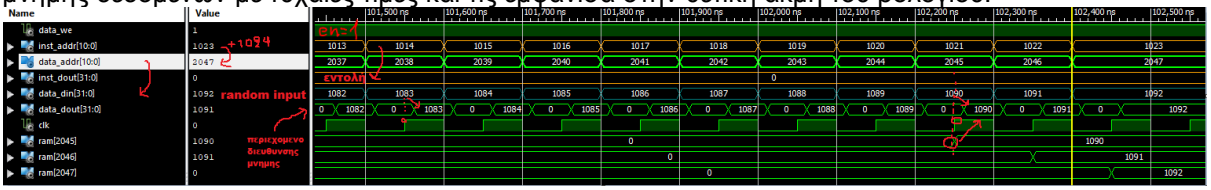


Όπως φαίνεται στο testbench ο PC προχωράει κατά 4 αλλά οι εντολές που βγάζει η μνήμη προχωράν κατά 1. Όταν το immed=4 προχωραει 2 θέσεις και όταν immed=8 προχωραει 3 θέσεις, με pc_sel=1.

RAM: Είναι η κύρια μνήμη, αποτελείται από 2048X32 μπιτ , περιέχει στο πρώτο μισό τις εντολές του προγράμματος και στο υπόλοιπο τα δεδομένα. Αρχικά παίρνει τις εντολές από το αρχείο rom.data και αρχικοποιεί με μηδενικά τις τιμές της θέσης μνήμης των δεδομένων. Έπειτα η RAM μπορεί να εμφανίσει το περιεχόμενο της διεύθυνσης που επιλέγουμε καθώς και να αποθηκεύει τιμές που εισάγουμε βάση επιλογής. Έχει δύο εξόδους, για την τιμή της διεύθυνσης εντολών και δεδομένων αντίστοιχα σε κάθε κύκλο ρολογιού.

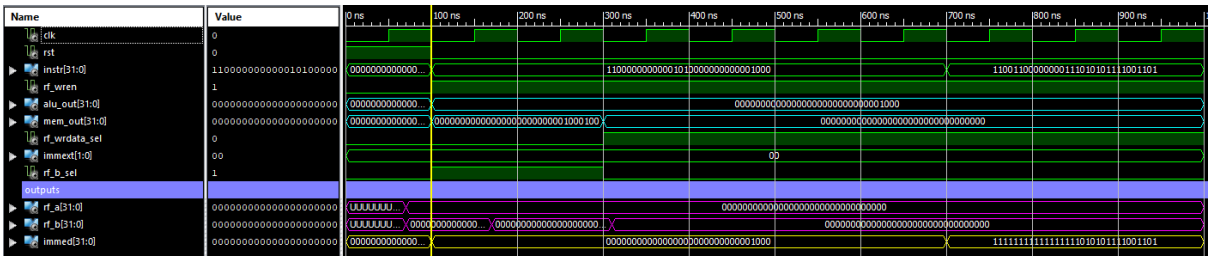


Για την προσομοίωση του κυκλώματος, με την χρήση for loop, γέμισα όλες τις θέσεις του κομματιού μνήμης δεδομένων με τυχαίες τιμές και τις εμφάνισα στην θετική ακμή του ρολογιού.



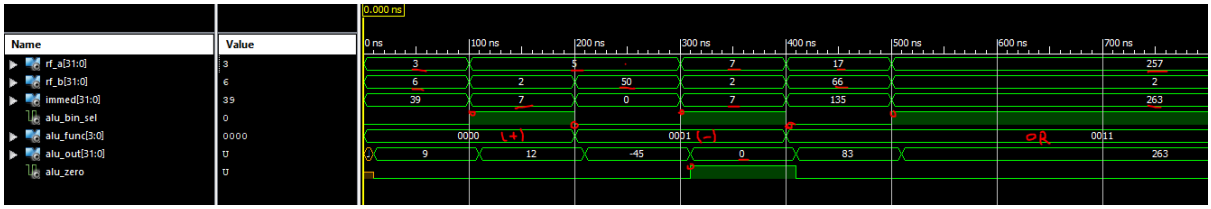
Επίσης εμφάνισα όλες τις εντολές που ήταν αποθηκευμένες στις διευθύνσεις της μνήμης από το rom.data, στο κομμάτι των εντολών.

DEC_STAGE: Σε αυτήν την βαθμίδα γίνεται η αποκωδικοποίηση των εντολών. Αποτελείται από την Register File και δύο πολυπλέκτες. Έναν για να επιλέξουμε αν τα δεδομένα που θα γραφτούν στην RF είναι από την μνήμη ή από την ALU και έναν mux για να διαλέξουμε το πώς θα γίνει το extension σε immidiate εντολές.

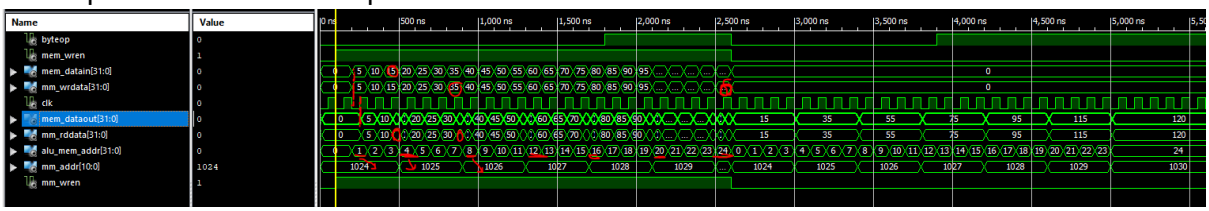


Από την μνήμη έρχεται η εντολή μήκους 32 μπιτ που πρέπει να εκτελεστεί. Χωρίζεται σε κομμάτια καθώς τμήματά της, περιέχουν διαφορετικές πληροφορίες ανάλογα με το τι τύπος εντολής είναι. Οι καταχωρητές της Register File γράφονται ανάλογα την εντολή, και των εισόδων επιλογής που έχει η βαθμίδα, και τα δεδομένα τους στέλνονται στην Exstage για να γίνει η πράξη.

EX_STAGE: Ή αλλιώς βαθμίδα εκτέλεσης εντολών, είναι ένα συνδυαστικό κύκλωμα που έχει ως σκοπό να διαλέξει ποιά πράξη της ALU θα εκτελεστεί ανάμεσα στον καταχωρητή RF_A και τον δεύτερο τελεσταίο που είναι RF_B για R-type εντολές, ενώ Immed για I-type εντολές, βάση επιλογής(ALU_Bin_sel).

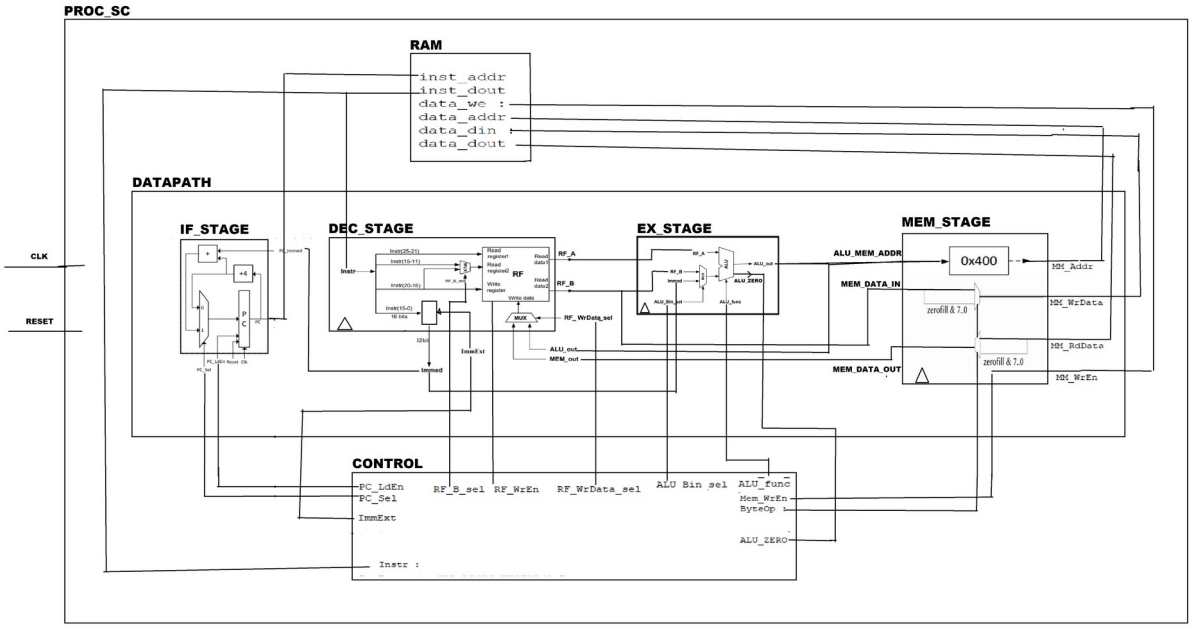


MEM_STAGE: Είναι η βαθμίδα που συνδέει τον επεξεργαστή με την μνήμη. Ο επεξεργαστής λειτουργεί με τιμές 32 bit ενώ η μνήμη με διευθύνσεις 10 bit. Για να γίνει η μεταξύ τους σύνδεση σε περιπτώσεις load/store, η mem_stage παίρνει απο τα 32 μπιτ τα 12-2 όπου υπάρχει η πληροφορία της διεύθυνσης μνήμης,και προσθέτει στο 10-μπιτο σήμα 1024 ώστε να πάει στο κομμάτι της μνήμης με τα δεδομένα. Επίσης για τις περιπτώσεις lb/sb κρατάει τα τελευταία 8 μπιτ και κάνει ZeroFill στα υπόλοιπα.



Στην προσομοίωση βλέπουμε να γίνεται εγγραφή στην μνήμη με τιμές πολλαπλάσια του 5, όταν η διεύθυνση είναι πολλαπλάσιο του 4. Έπειτα εμφανίζω τις τιμές που γράφτηκαν στην μνήμη.

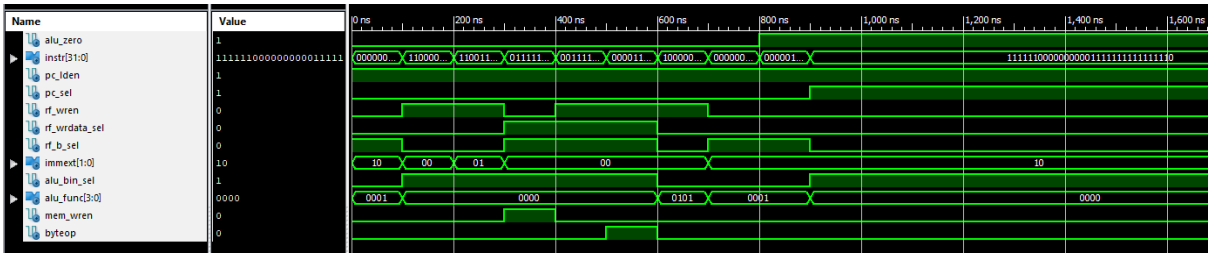
ΤΡΙΤΗ ΦΑΣΗ



Παραπάνω βλέπουμε το σχηματικό διάγραμμα του επεξεργαστή, με τα υποκυκλώματά του καθώς και τις συνδέσεις αυτών μεταξύ τους.

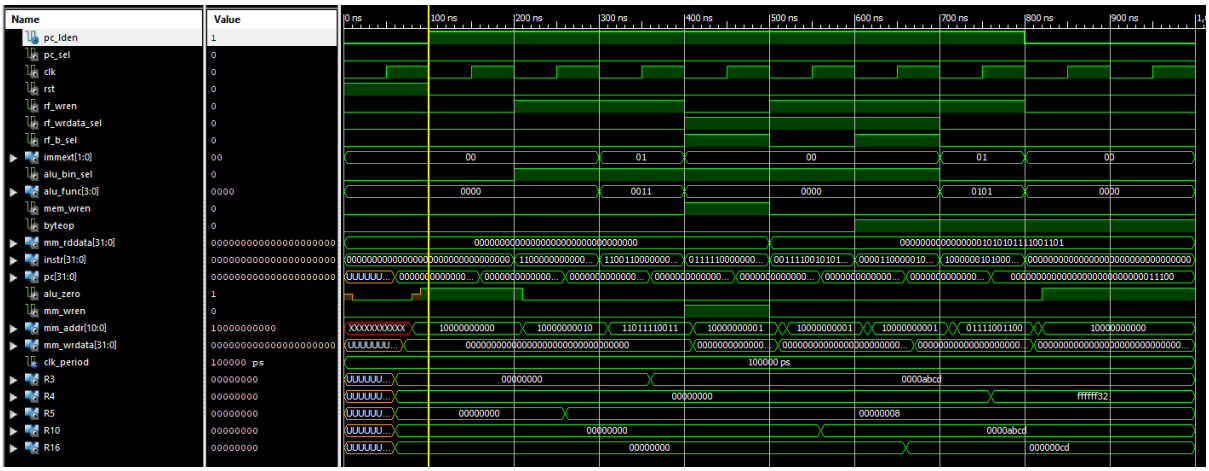
Σε αυτή την φάση έγινε όλη η μαγεία. Συνδέσαμε όλα τα δομικά στοιχεία του επεξεργαστή μας, περάσαμε ένα αρχείο .rom που μας επιτρέπει να τρέχουμε εντολές και τις είδαμε να εκτελούνται !

CONTROL: Είναι ουσιαστικά ένας μεγάλος πίνακας αληθείας. Παίρνει από την μνήμη ως είσοδο την τιμή της εντολής που πρέπει να εκτελεστεί και βγάζει ως έξοδο τα σήματα για τις επιλογές που πρέπει να γίνουν σε κάθε βαθμίδα. Έχει επίσης ως είσοδο την ALU_ZERO από την οποία κάνω έλεγχο ισότητας για τις εντολές beq και bne στους καταχωρητές της εντολής.



Στην προσομοίωση βλέπουμε τις τιμές που παίρνουν τα σήματα ελέγχου για τις εντολές του προγράμματος αναφοράς 1 επιβεβαιώνοντας την σωστή λειτουργία του Control.

DATAPATH: Είναι η βαθμίδα όπου γίνεται η σύνδεση μεταξύ IF,EX,DEC,MEM και δημιουργούνται τα σήματα που που θα συνδεθούν αργότερα με το CONTROL και RAM. Φτιάχνω ουσιαστικά τα καλώδια που συνδέουν όλες τις μονάδες μεταξύ τους.

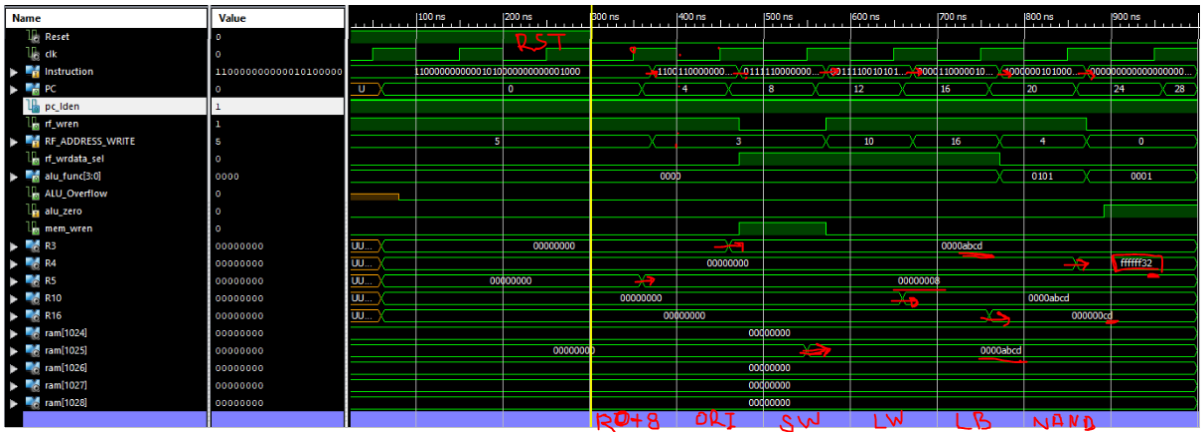


Για το simulation του datapath έκανα οτι και στο πρόγραμμα αναφοράς 1 που μας δόθηκε για το Proc_SC επιβεβαιώνοντας την σωστή λειτουργία του κυκλώματος.

Πρόγραμμα αναφοράς #1

```
00: addi r5, r0, 8
04: ori  r3, r0, 0xABCD
08: sw   r3, 4(r0)
0C: lw   r10, -4(r5)
10: lb   r16, 4(r0)
14: nand r4, r10, r16
```

PROC_SC: Είναι το TOP LEVEL του επεξεργαστή. Συνδέει στο DATAPATH το CONTROL και τη RAM. Έχει ως σήματα μόνο reset και clock καθώς όλα τα υπόλοιπα σήματα δημιουργούνται και χρησιμοποιούνται εντός της μονάδας αυτής.

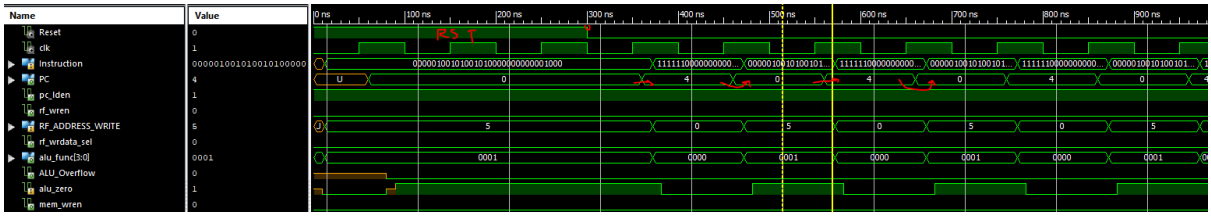


Στο πρώτο simulation βλέπουμε το πρόγραμμα αναφοράς 1, οι τιμές που βλέπουμε είναι για κάθε πράξη σωστές, επιβεβαιώνοντας σωστή λειτουργία του κυκλώματος.

Πρόγραμμα αναφοράς #2

```
// *** Δεύτερο πρόγραμμα, μόνο διακλαδώσεις, εκτελεί δύο εντολές με αποτυχημένη διακλάδωση και ξανά
00: bne r5, r5, 8          // αποτυχημένη διακλάδωση
04: b -2                   // branch (PC=04 + 4 -2*4 = 00) infinite loop!
08: addi r1, r0, 1         // δεν θα εκτελεστεί
```

Στο δεύτερο simulation βλέπουμε το πρόγραμμα αναφοράς 2



ΒΙΒΛΙΟΓΡΑΦΙΑ:

-e-class.gr Υλικό Εργαστηρίου/Διαλέξεις
-xilinx.com
-stackoverflow.com