ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ & ΥΛΙΚΟΥ
ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ ΓΙΑ ΤΟ ΜΑΘΗΜΑ:
Οργάνωση Υπολογιστών
ΗΡΥ 302
http://www.mhl.tuc.gr
ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2024

Αναφορά Εργαστηριακής Άσκησης 1

Αριθμός Ομάδας:66

Λαμπράκης Μιχάλης 2020030077
Δήμας Χρήστος 2021030183

1.Σκοπός της Άσκησης

Ο σκοπός της Άσκησης αυτής είναι η σχεδίαση ενός πλήρους λειτουργικού επεξεργαστή μονο κύκλου με την χρήση της γλώσσας VHDL. Η σχεδίαση χωρίστηκε σε 3 φάσεις με σκοπό την πλήρης κατανόηση της λειτουργίας του επεξεργαστή.

Στην $1^η$ φάση σχεδιάστηκε μια μονάδα αριθμιτικών και λογικών πράξεων (ALU) καθώς και ένα αρχείο καταχωρητών(Register File).

Στην $2^η$ σχεδιάστηκαν οι βασικές βαθμίδες του Datapath του επεξαργαστή. Αυτές είναι βαθμίδα ανάκλησης εντολών(IFSTAGE), αποκωδηκοποίησης εντολών(DECSTAGE), εκτέλεσης εντολών(EXSTAGE) και πρόσβασης μνήμης (MEMSTAGE). Για την σχεδίαση χρειάστηκαν η (ALU) και ο (Register file) από την προηγούμενη φάση καθώς και με την χρήση του εργαλείου Xillinx Core Generator υλοποιήθηκαν 2 μνήμες, μια ROM 1024x32 για την αποθήκευση των βασικών εντολών και μια κύρια μνήμη RAM 1024x32.

Τέλος, στην $3^η$ φάση συνδέθηκαν τα επιμέρους στοιχεία για να κατασκευαστεί το ενιαίο DATAPATH καθώς και υλοποιήθηκε το CONTROL του επεξεραστή το οποίο παράγει τα σωστά σήματα ελέγχου για την κάθε εντολή.

2.Περιγραφή της Σχεδίασης

$1^η$ φάση

Στην $1^η$ φάση αρχικά σχεδιάστηκε η μονάδα αριθμιτικών και λογικών πράξεων (ALU) η οποιά παίρνει σαν είσοδο δύο 32bit τελεστέους και έναν 4bit κωδικό πράξης. Ανάλογα με τον κωδικό, η ALU βγάζει στην έξοδο το αποτέλεσμα της πράξης καθώς και 3 διαφορετικά flags που επισημαίνουν αν το αποτέλεμσα της πράξης είναι α) 0, β)έχει κρατούμενο εξόδου 1, γ)έχει υπερχείληση. Όσον αφορά την υπερχείληση, μπορεί να προκύψει μόνο στην πρόσθεση και στην αφαίρεση. Στην πρόσθεση, υπάρχει υπερχείληση αν και οι δύο προσθετέοι είναι ομόσιμοι μεταξύ τους και ετερόσημοι με το αποτέλεσμα, ενώ στην αφαίρεση έχουμε άν ο μειώτέος είναι είναι θετικός και ο αφαιρετέος αρνητικός και η διαφορά αρνητική η το ανάποδο. Για την υλοποίηση των πράξεων χρησιμοποιήθηκαν οι βιβλιοθήκες IEEE.NUMERIC_STD.ALL και IEEE.STD_LOGIC_SIGNED.ALL ώστε να μην χρειαστεί να υλοποιηθούν από την αρχή τα κυκλώματα για τις πράξεις.

Στη συνέχεια σχεδιάστηκαν τα component τα οποία αποτελούν το αρχείο καταχωρητών (register file). Αρχικά σχεδιάστηκε ένας σύγχρονος καταχωρητς 32bit με Reset και WriteEnable. Στη συνέχεια ένας πολυπλέκτης ο οποίος ανάλογα με το σήμα sel (5bit) επιλέγει ποιά απο τις 32 εισόδος θα περάσει στην έξοδο και τέλος ένας αποκωδικοποιητής που ανάλογα με το σήμα εισόδου awr (5bit) επιλέγει ποιό απο τα 32bit της εξόδου θα είναι 1.

Το αρχείο καταχωρητών περίεχει 2 εισόδους ασύγχρονης ανάγνωσης που ελέγχουν 2 πολυπλέκτές και καθορίζουν τις εξόδους που θα διαβαστούν, 1 είσοδο σύγχρονης εγγραφής που μέσω του αποκωδικοποιητή καθορίζει σε ποιόν καταχυρητή θα γράψουμε, την είσοδο των δεδομένων, τα σήματα ελέγχου Reset και WriteEnable και το ρολόι. Χρειάστηκαν 32 καταχωρητές οι οποίοι δημιουργήθηκαν μέσω for-generate. Από αυτούς η τιμή του R0 είναι πάντα 0.

$2^{η}$ φάση

Η $2^{η}$ φάση αποτελείται από 4 στάδια. Αρχικά το $1^{ο}$ στάδιο είναι η βαθμίδα ανάκληση εντολων (IF Stage). Σε αυτό το στάδιο διαβάζεται μια εντολή από την μνήμη (χρησιμοιούμε μια Distributed ROM) και την φορτώνουμε σε έναν καταχωρητή, τον Program Counter, ο οποίος σε κάθε κύκλο περιέχει την εντολή που εκτελείται την συγκεκριμένη χρονική στιγμή. Στον επόμενο κύκλο ο PC προχωράει στην επόμενη εντολή και παίρνει την τιμή (PC+4) εκτός αν η εντολή είναι brunch, brunch equal ή brunch not equal όπου τότε ο PC παίρνει την τιμή (PC+4+Immediate) και τότε η εκτέλεση του προγράμματος μεταφέρεται στην διεύθυνση αυτή. Η μνήμη λαμβάνει ως είσοδο τα [11:2] bits της εξόδου του καταχυρητή αφού αυτή είναι οργανωμένη σε words. Έτσι όσο αυξάνεται ο PC κατά 4 η μνήμη θα αυξάνεται κατά 1.

Το $2^{ο}$ στάδιο αποτελεί η βαθμίδα αποκωδικοποίησης εντολών (DEC Stage). Σε αυτό το στάδιο η εντολή που φορτώνεται από την μνήμη αποκωδικοποιείται, με σκοπό την εγγραφή και προσπέλαση της στο αρχείο καταχωρητών. Επιπλέον, γίνεται και η κατάλληλη επεξεργασία της τιμής Immediate. Ο καταχωρητής ανάγνωσης 1 είναι πάντα ο rs ενώ ο καταχωρητής 2 είναι είτε ο rt είτε ο rd. Ένα πολυπλέκτης καθορίζει ποιός από τους 2 θα περάσει, όπου εκεί πρακτικά γίνεται και ο διαχορισμός των R-Type από τις I-Type εντολές. Ο καταχωρητής εγγραφής είναι πάντα ο rd. Ακόμη η επεξεργασία του Immediate γίνεται στην μονάδα "Cloud" όπου ανάλογα με την τιμή του σήματος CloudControl γίνεται 1.Zero Fill, 2.Sign Extend, 3.Zero Fill & Shift και 4.Sign Extend & Shift 2. Τέλος, η είσαγωγή των δεδομένων προς εγγραφή στο Register File επιλέγεται μέσω ενός άλλου πολυπλέκτη να είναι είτε από την μνήμη RAM σε περίπτωση των εντολών load και store είτε απο την έξοδο της ALU στις υπόλοιπες εντολές.

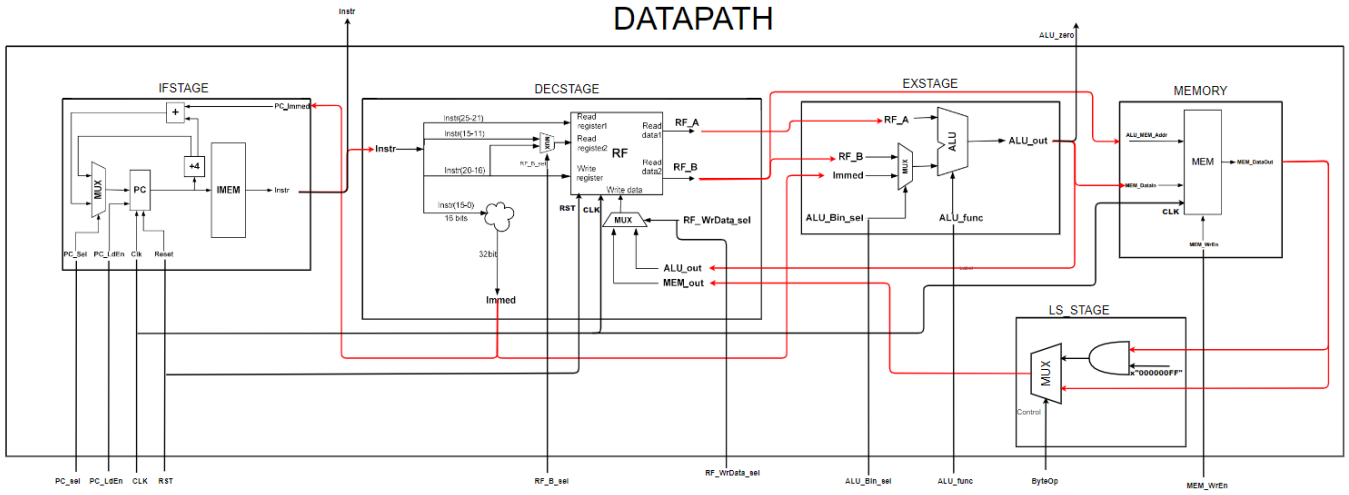
Το $3^{ο}$ στάδιο είναι η βαθμίδα εκτέλεσης εντολών (EX Stage). Σε αυτό το στάδιο γίνεται η εκτέλεση της εντολής και ο υπολογισμός του αποτελέσματος της ALU. Το $1^{ο}$ σήμα εισόδου της αποτελούν τα δεδομένα ανάγνωσης του καταχωρητή 1(έξοδος απο Register File) και το $2^{ο}$ μέσω ενός πολυπλέκτη είναι είτε τα δεδομένα ανάγνωσης του καταχωρητή 2 είτε το επεξεργασμένο Immediate.

Το $4^{ο}$ και τελευταίο στάδιο αποτελεί η βαθμίδα πρόσβασης μνήμης (MEM Stage). Σε αυτό το στάδιο υλοποιήθηκε μια μνήμη RAM 1024 θέσεων των 32bit. Η μνήμη είναι Read First και έχει μια θύρα ανάγνωσης και εγγραφής.

Τέλος για την υλοποίηση της εντολής load byte έχει δημιουργήθεί ένα LS Stage το οποίο παίρνει σαν είσοδο την έξοδο της μνήμης RAM και μέσω ενός πολυπλέκτη επιλέγει εάν θα βγει αυτούσια στην έξοδο είτε αν ,στην περίπτωση της εντολής load byte, γίνει Zero Fill στα bit [31:8].

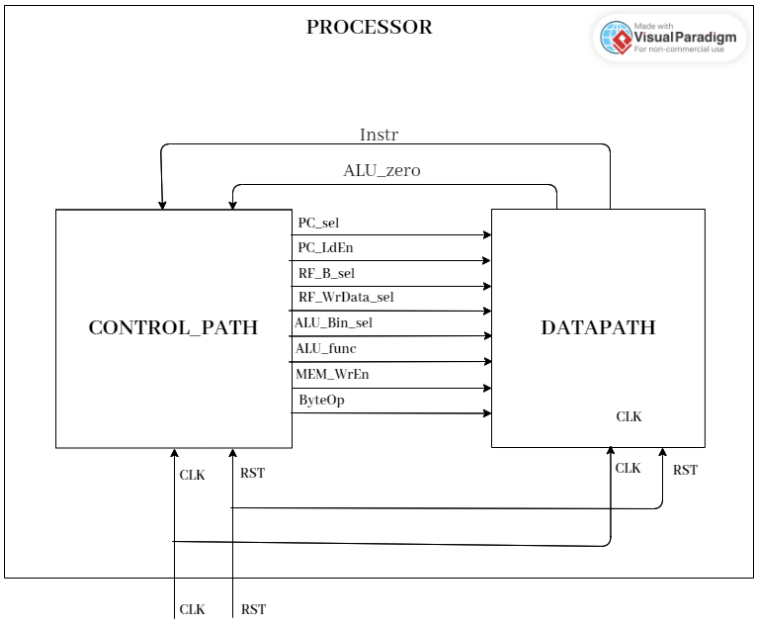
3η φάση

Στην 3η φάση ενώθηκαν τα 4 στάδια και υλοποιήθηκε το DATAPATH το οποίο πρακτικά είναι υπεύθυνο για να πραγματοποιήσει όλες τις λειτουργίες του επεξεργστή, ανάλογα με τις εντολές που εισάγονται.



Στην συνέχεια δημιουργήθηκε το CONTROL το οποίο είναι υπεύθυνο να στέλνει τα κατάλληλα σήματα ελέγχου στο DATAPATH ανάλογα με την εντολή που εκτελείται. Παίρνει σαν εισόδους ένα ασύγχρονο Reset καθώς και το flag Zero της ALU το οποίο χρησιμοποιείται στις εντολές διακλαδώσεων brunch και παράγει εξόδους για όλα τα σήματα εισόδου του DATAPATH.

Το DATAPATH και το CONTROL ενώθηκαν σε ένα Top Level αρχείο PROCESSOR το οποίο τρέχει και επιβεβαιώνει την ορθή λειτουργία του επεξεργστή. Δεν έχει εισόδους και εξόδους πέρα από το Reset και το ρολόι.

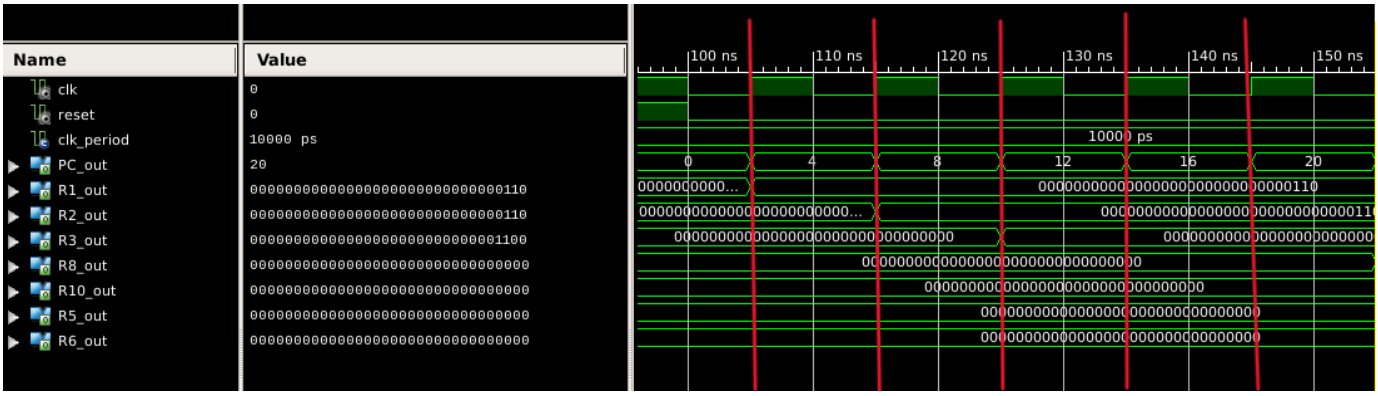


3.Αναφορά αποτελεσμάτων – Επιβεβαίωση λειτουργίας

Παρακάτω παρουσιάζονται οι κυματομορφές του επεξεργαστή που επιβεβαιώνουν την λειτουργία του με όλα τα πιθανά Instructions. Οι εντολές φορτώθηκαν στην μνήμη IMEM μέσω .coe αρχείων.

Αρχικά το πρώτο σετ εντολών, το οποίο δώθηκε, βρίσκεται στο αρχείο Initialization.coe και περιέχει τις εξής εντολές:

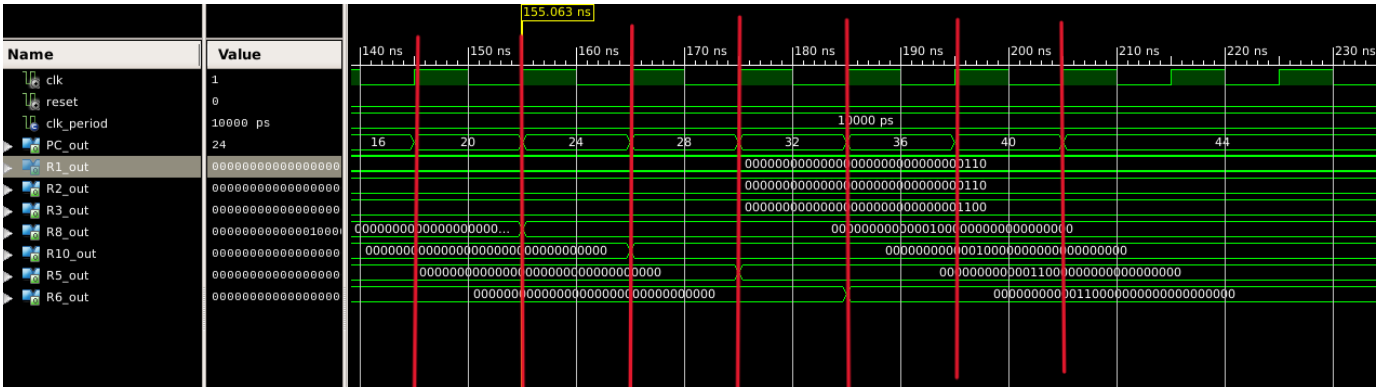
1.li r1,6
2.li r2,6
3.add r1,r3,r2
4.bne r1,r2, 0
5.beq r1,r2, 0



Για να επιβεβαιώσουμε την λειτουργία του επεξεργαστή βλέπουμε τις εξόδους των καταχωρητών στην θετική ακμή του ρολογιού. (Όταν δηλαδή περνάνε τα αποτελέσματα στην έξοδο). Αρχικά το σήμα Reset είναι ενεργό οπότε όλες οι εξόδοι μηδενίζονται. Έπειτα εκτελούνται σειριακά οι παραπάνω εντολές (ο PC αυξάνεται κατά 4 σε κάθε κύκλο) και έχουμε τα εξής αποτελέσματα (με κόκκινο χρώμα φαίνεται ο κάθε κύκλος):
1. Καταχωρείται στον R1 το 6.
2. Καταχωρείται στον R2 το 6.
3. Στον R3 καταχωρείται το αποτέλεσμα του αθροίσματος των R1 και R2.
4. Τα R1 και R2 είναι διαφορετικά οπότε η εκτέλεση του προγράμματος συνεχίζεται κανονικά. 5. Τα R1 και R2 είναι ίσα οπότε η εκτέλεση του προγράμματος πρέπει να μεταφερθεί στην θέση PC+4+Immediate (στην περίπτωση μας 0) οπότε συνεχίζεται κανονικά.

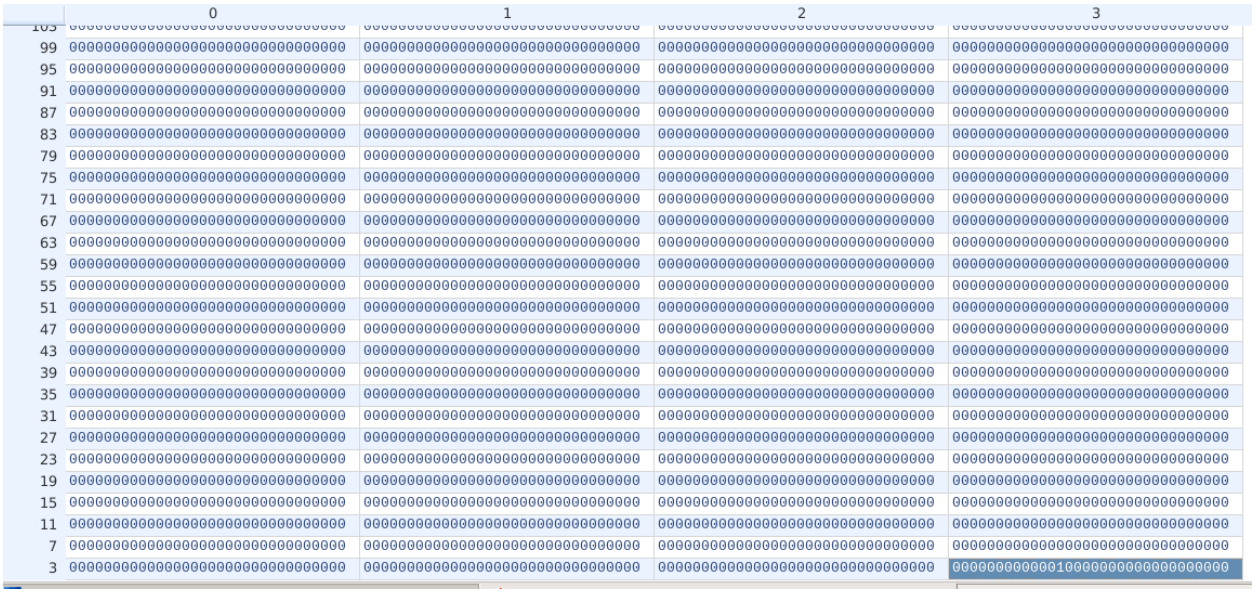
Συνεχίζουμε με τις εντολές:

1. lui r8,4
2. lui r10,16
3. or r8,r5,r10
4. rol r5,r6,1
5. ror r6,r5,1
6. sw r5,4(r10)



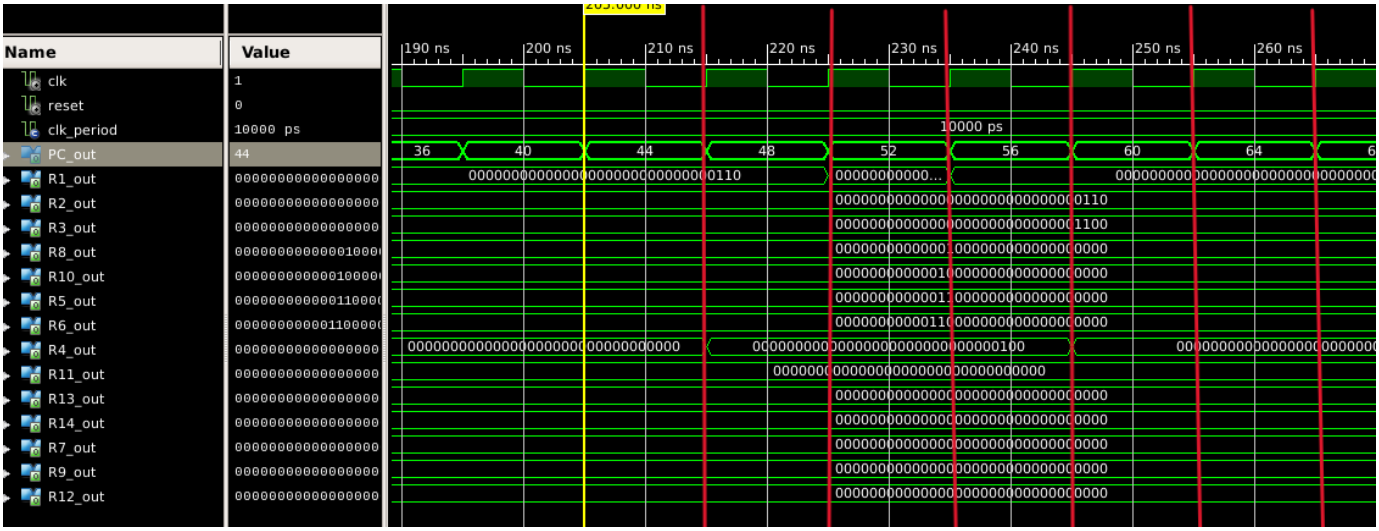
Στο παραπάνω σετ εντολών βλέπουμε τα εξής:

1. Στον R8 καταχωρείται το 4.
2. Στον R10 καταχωρείται το 16.
3. Στον R5 καταχωρείται το αποτέλεσμα του OR των R8 και R10.
4. Ο R5 γίνεται Rotate Left κατά 1 και αποθηκεύεται το αποτέλεσμα στον R6.
5. Ο R6 γίνεται Rotate Right κατά 1 και αποθηκεύεται το αποτέλεσμα στον R5.
6. Αποθηκεύει την τιμή του 10 στην διεύθυνση [τιμήR5+4] όπου αυτο βγαίνει στην πρώτη θέση της μνήμης πράγμα που επιβεβαιώνεται αμα ανοίξουμε την μνήμη όπως φαίνεται παρακάτω.



Στην συνέχεια επεκτάθηκε το αρχείο Initialization.coe ώστε να επιβεβαιωθεί και η λειτουργία των υπόλοιπων εντολών. Οι εντολές που προστέθηκαν είναι:

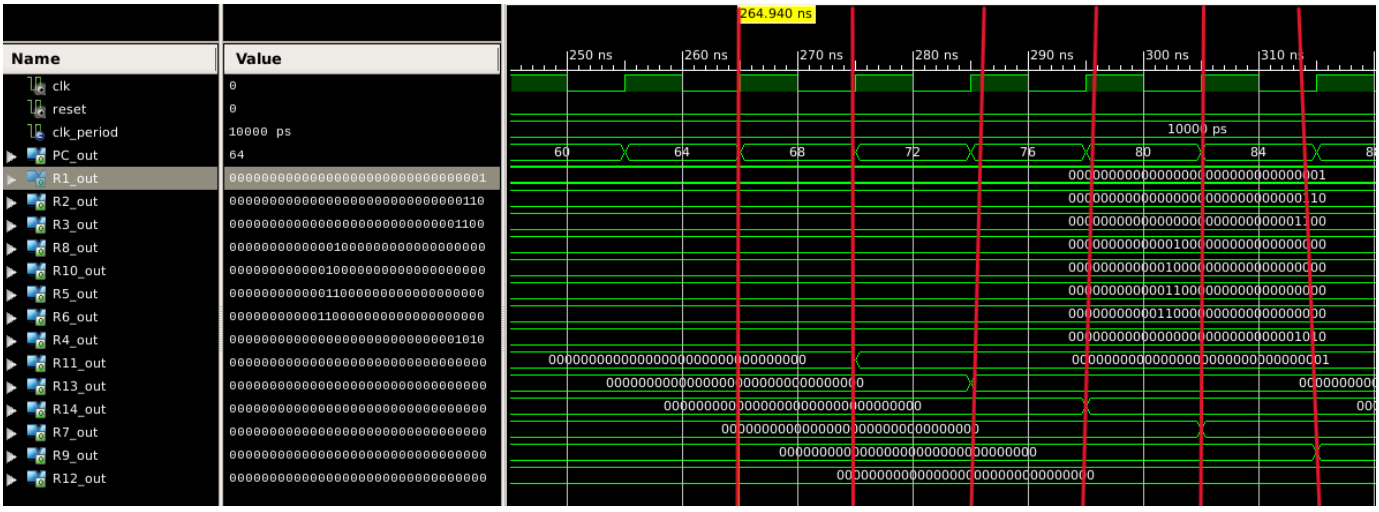
1. and r1,r4,r3
2. sub r2,r1,r4
3. sra r1,r1
4. addi r2,r4,4
5. beq r4,r3,0
6. bne r4,r3,0



Στο παραπάνω σετ εντολών βλέπουμε τα εξής:
1. Στον R4 καταχωρείται το αποτέλεσμα του AND των R1 και R3.
2. Στον R1 καταχωρείται το αποτέλεσμα της αφαίρεσης των R2 και R4.
3. Ο R1 γίνεται Shift Right Arithmetic κατά 1.
4. Στον R4 καταχωρείται το αποτέλεσμα της πρόσθεσης του R2 και του Immediate.
5. Τα R4 και R3 είναι ίσα οπότε η εκτέλεση του προγράμματος συνεχίζεται κανονικά.
6. Τα R4 και R3 είναι διαφορετικά οπότε η εκτέλεση του προγράμματος πρέπει να μεταφερθεί στην θέση PC+4+Immediate (στην περίπτωση μας 0) οπότε συνεχίζεται κανονικά.

Συνεχίζουμε με τις εντολές:

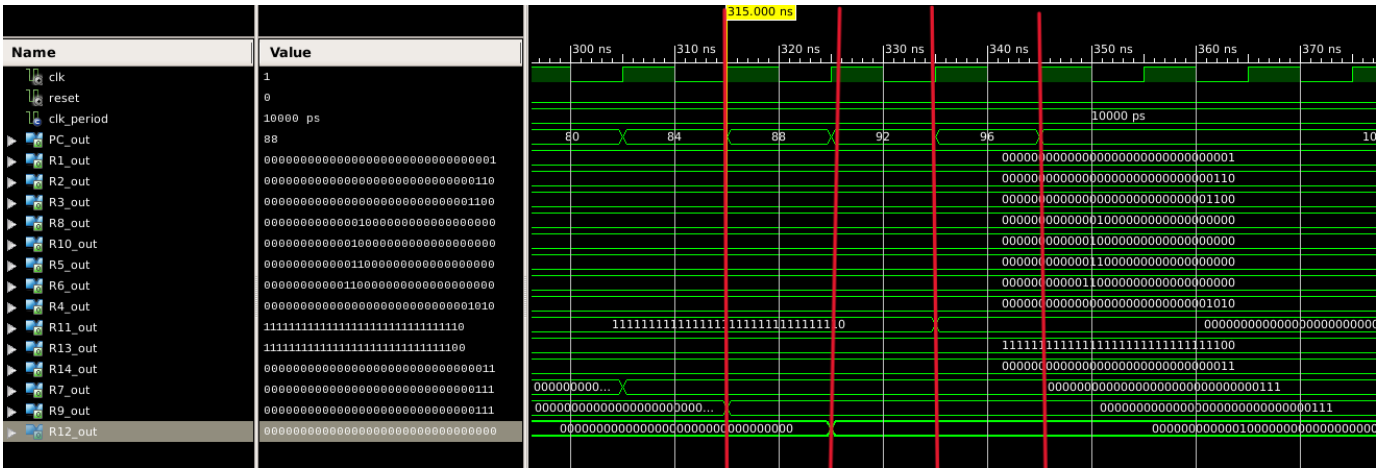
1. not r1,r11
2. sll r11,r13
3. srl r2, r14
4. addi r14,r7,4
5. ori r7,r9,1



Βλέπουμε τα εξής:
1. Στον R11 καταχωρείται το αποτέλεσμα του NOT του R1.
2. Ο R11 γίνεται Shift Left Logical κατά 1 και το αποτέλεσμα αποθηκεύεται στον R13.
3. Ο R2 γίνεται Shift Right Logical κατά 1 και το αποτέλεσμα αποθηκεύεται στον R14.
4. Στον R7 καταχωρείται το αποτέλεσμα της πρόσθεσης του R14 και του Immediate.
5. Στον R9 καταχωρείται το αποτέλεσμα του OR των R7 και του Immediate.

Και τέλος προσθέσαμε τις εντολές:

1. lw r5,4(r12)
2. lb r5,4(r11)
3. b 1



Και τελικά βλέπουμε τα εξής:
1. Στον R12 καταχωρείται η τιμή της μνήμης MEM[0000_00001]=R10
2. Στον R11 καταχωρείται η τιμή της μνήμης MEM[R5+4]=0
3. Η εκτέλεση του προγράμματος μεταφέρεται στην θέση PC+4+4 που επιβεβαιώνεται αφού PC=104.

4.Συμπεράσματα

Αυτή η άσκηση είναι μια πολύ καλή εξάσκηση για την κατανόηση της λειτουργίας ενός επεξεργαστή ενός κύκλου.Εντρυφήσαμε στην VHDL, την κατανοήσαμε και μάθαμε να χειριζόμαστε τις βιβλιοθήκες της για την διευκόλυση της υλοποίησης των κυκλωμάτων. Μέσω εξαντλητικών δοκιμών (testbenches) καταφέραμε να επιβεβαιώσουμε την λειτουργία του επεξεργαστή και κατανοήσαμε πόσο σημαντική είναι αυτή η διαδικασία για την αποφυγή λαθών και την εξοικονόμηση χρόνου.Τέλος, μάθαμε πως να κάνουμε ιεραρχική σχεδίαση, να χωρίζουμε τα sub modules και να τα συνδέουμε σε ένα Top Level αρχείο.