ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
Οργάνωση Υπολογιστών

Ομάδα: LAB31235453
Μπαλαμπάνης Ηλίας 2014030127
Μποκαλίδης Αναστάσιος 2014030069

Αναφορά Εργαστηρίου 3

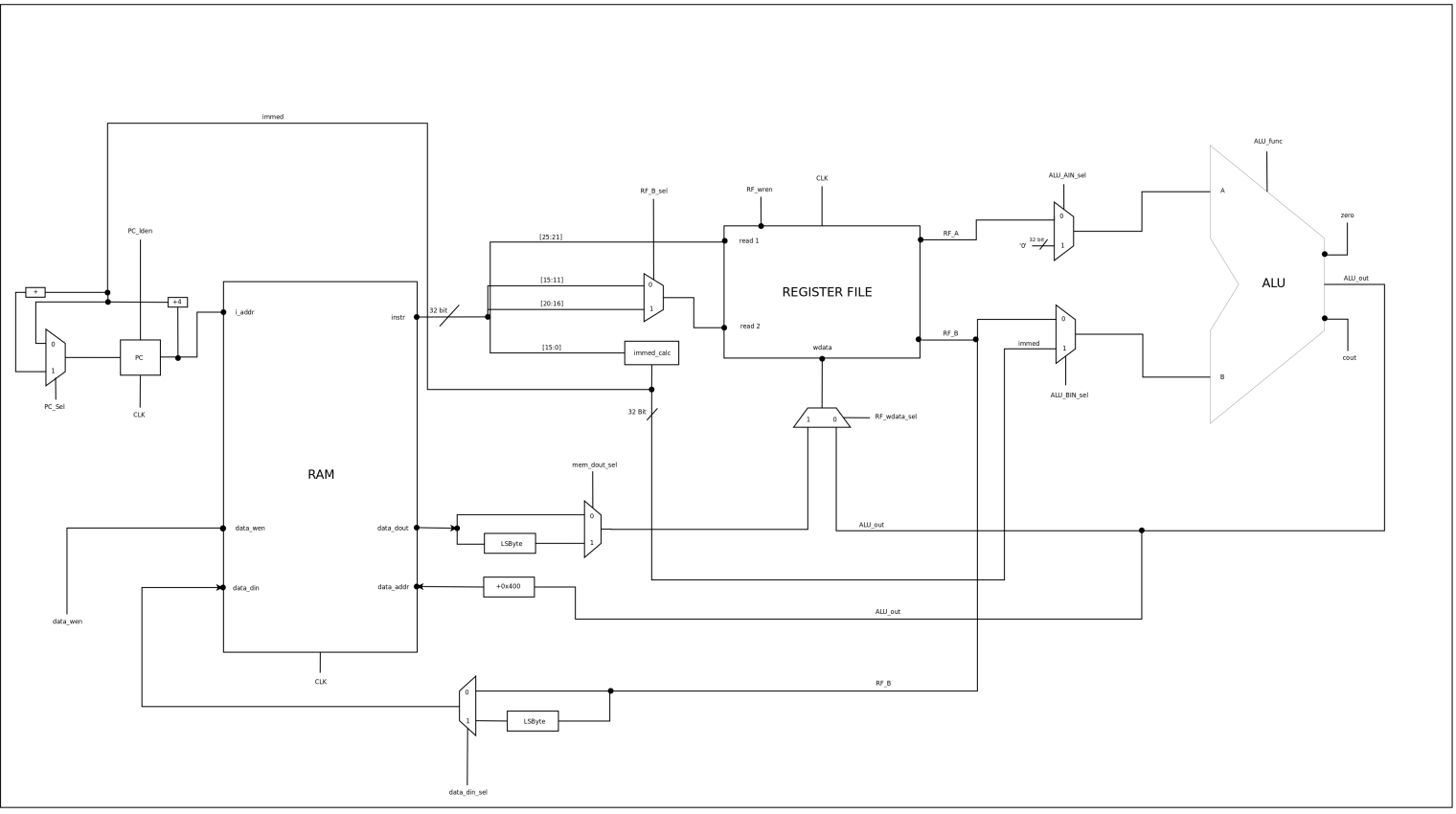
Σημείωση: Όλες οι εικόνες παρατίθενται με το .zip αρχείο της αναφοράς

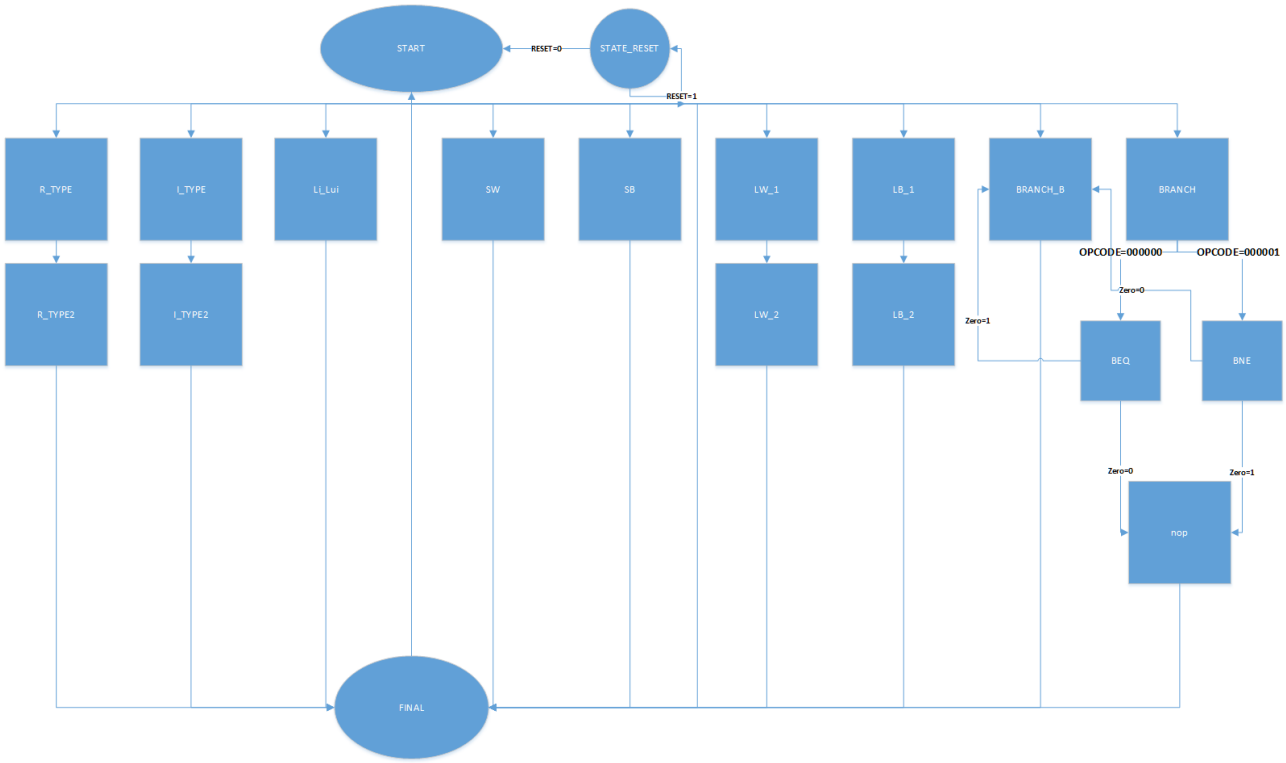
Περιγραφή Άσκησης

Μελετήσαμε όλα τα modules που υλοποίησαμε στα δύο προηγούμενα εργαστήρια και τα συνδέσαμε κατάλληλα έτσι ώστε να κατασκευάσουμε ένα ενιαίο datapath το οποίο φαίνεται παρακάτω.

Το datapath σε **block diagram**:

DATAPATH





Η fsm μας ξεκινά από την κατάσταση state_reset στην οποία όσο δίνουμε reset=1 μένουμε σε αυτήν την κενή κατάσταση και ενεργοποιείται το reset του PC, δηλαδή στην έξοδο του έχουμε μηδέν. Με το που δώσουμε reset=0 τότε μεταβαίνουμε στη κατάσταση START στην οποία διαβάζουμε τον opcode και το function από τις εντολές που μας έρχονται από την μνήμη και αναλόγως επιλέγουμε σε ποια κατάσταση-πράξη θα μεταβούμε.

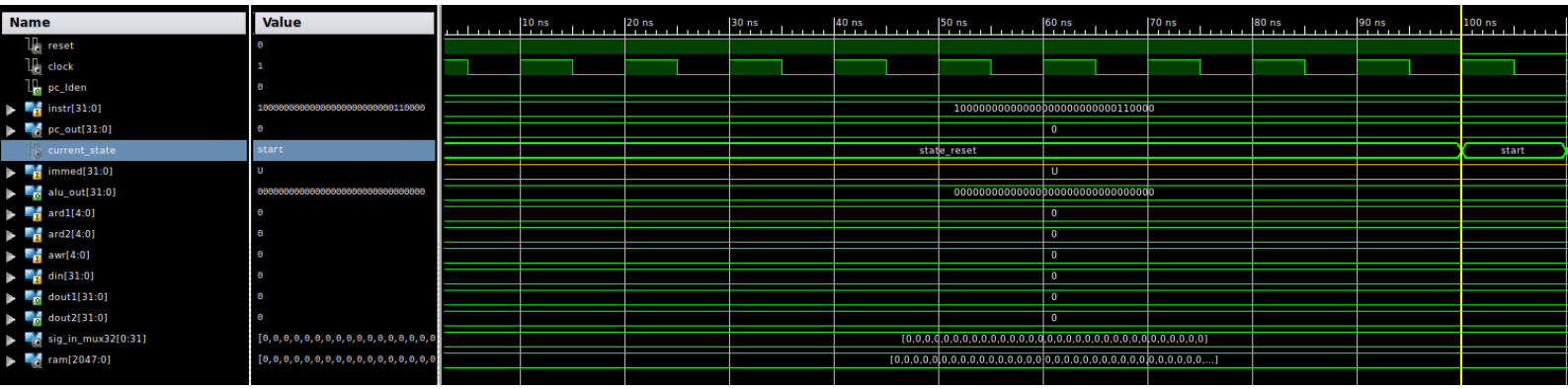
- **R_Type :** Σε αυτήν την κατάσταση αρχικά διαλέγουμε ποιό καταχωρητή θέλουμε να διαβάζουμε και σε ποιόν να γράψουμε δεδομένα. Επειδή η μονάδα των καταχωρητών, δηλαδή η RegisterFile, είναι σύγχρονη θα αναγκαστούμε να προσθέσουμε άλλη μια κατάσταση. Στην πρώτη διαβάζουμε τα δεδομένα από τους καταχωρητές και στην επόμενη κάνουμε την πράξη στην ALU και αποθηκεύουμε το αποτέλεσμα στον καταχωρητή που ορίζει το Instruction.
- **I_Type :**Αυτή η κατάσταση είναι παρόμοια με την προηγούμενη, απλώς η μόνη διαφορά είναι ότι αντί να διαβάζουμε ένα 2^o καταχωρητή διαβάζουμε το immed που έχουμε. Αυτό το πετυχαίνουμε με την κατάλληλη ρύθμιση των σημάτων ελέγχου των πολυπλεκτών.
- **Li_Lui :** Αυτή η κατάσταση είναι παρόμοια με την I_type μονό που τώρα δεν διαβάζουμε από καταχωρητή αλλά παίρνουμε το immed το προσθέτουμε με τον μηδέν στην ALU και το αποτέλεσμα το αποθηκεύουμε στον επιθυμητό καταχωρητή. Έτσι στην ουσία γίνεται η αποθήκευση μιας σταθεράς σε ένα καταχωρητή. Επίσης άλλη μια διαφορά που υπάρχει με τις προηγούμενες καταστάσεις είναι ότι σε αυτήν την εντολή χρειαζόμαστε ένα κύκλο λιγότερο επειδή δεν διαβάζουμε από τους καταχωρητές.

- **SW :** Σε αυτήν την κατάσταση κάνουμε παρόμοια βήματα με την li_lui απλώς αντί να αποθηκεύουμε το αποτέλεσμα της ALU σε καταχωρητή την αποθηκεύουμε στην μνήμη. Σημαντικό σε αυτήν την περίπτωση είναι να αποθηκεύουμε στη σωστή θέση της μνήμης το αποτέλεσμα που θέλουμε. Αυτό το επιτυγχάνουμε με το να προσθέτουμε στην έξοδο της ALU το 0x400 και αυτό να πηγαίνει στην είσοδο ανάγνωσης διεύθυνσης της μνήμης RAM που φτιάξαμε.
- **SB :** Είναι η ίδια κατάσταση με την προηγούμενη μόνο που σε αυτήν την περίπτωση επιλέξουμε μέσω ενός καταχωρητή να αποθηκεύσουμε το αποτέλεσμα που θέλουμε σε bytes και όχι σε word.
- **LW :** Σε αυτήν την κατάσταση θέλουμε να φορτώσουμε σε ένα καταχωρητή μια τιμή από την μνήμη. Γι΄αυτό χρειαζόμαστε 2 κύκλους. Ένα κύκλο όπου θα βγάλει την τιμή από την μνήμη και έναν που θα γράψει την τιμή στον επιθυμητό καταχωρητή. Επίσης στον $2^{ο}$ κύκλο θα αλλάξει και η διεύθυνση όπου θέλουμε να διαβάσουμε από την μνήμη έτσι ώστε να είναι έτοιμο να διαβάσει νέα δεδομένα σε αντίστοιχη εντολή.
- **LB :** Αυτή η κατάσταση είναι ίδια με την προηγούμενη μόνο που στην έξοδο της μνήμης επιλέγουμε τα δεδομένα που θα βγουν να είναι σε bytes και όχι σε word. Αυτό επιτυγχάνεται με το κατάλληλο σήμα ελέγχου ενός πολυπλέκτη.
- **Branch_B :** Σε αυτή τη κατάσταση στην ουσία αυξάνουμε τον PC σύμφωνα με το αποτέλεσμα της PC+4+Immed.
- **Branch :** Μέσω αυτής την κατάστασης ελέγχουμε πότε χρειαζόμαστε να κάνουμε BEQ ή BNE
- **BEQ :** Ελέγχουμε αν 2 καταχωρητές είναι ίσοι. Αυτό το υλοποιούμε αφαιρώντας τις τιμές στην ALU. Αν στην έξοδο της ALU έχουμε **Zero=1** δηλαδή οι καταχωρητές είναι ίσοι, εκτελούμε την εντολή **branch_b.** Διαφορετικά μεταβαίνουμε στην κατάσταση **nop** στην οποία απλώς αυξάνουμε τον PC κατά 4 έτσι ώστε να πάρουμε την νέα εντολή.
- **BNE :** Στην ουσία αυτή η εντολή είναι η αντίστροφη της BEQ. Αν έχουμε **Zero=0** εκτελούμε την εντολή **Branch_B.** Αντιθέτως μεταβαίνουμε στην κατάσταση **nop.**
- **Nop :** είναι μια κενή κατάσταση στην οποία ενεργοποιούμε το PC_LoadEnable και αυξάνουμε τον PC κατά 4.
- ***FINAL :*** *είναι μια κενή κατάσταση στην οποία περιμένουμε να έρθει στον επόμενο κύκλο το νέο Instruction.*

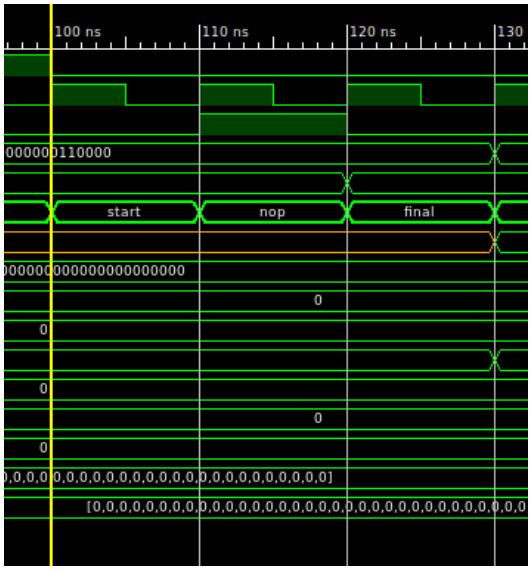
Κυματομορφές

Και με βάση το πρόγραμμα που μας δόθηκε έχουμε:

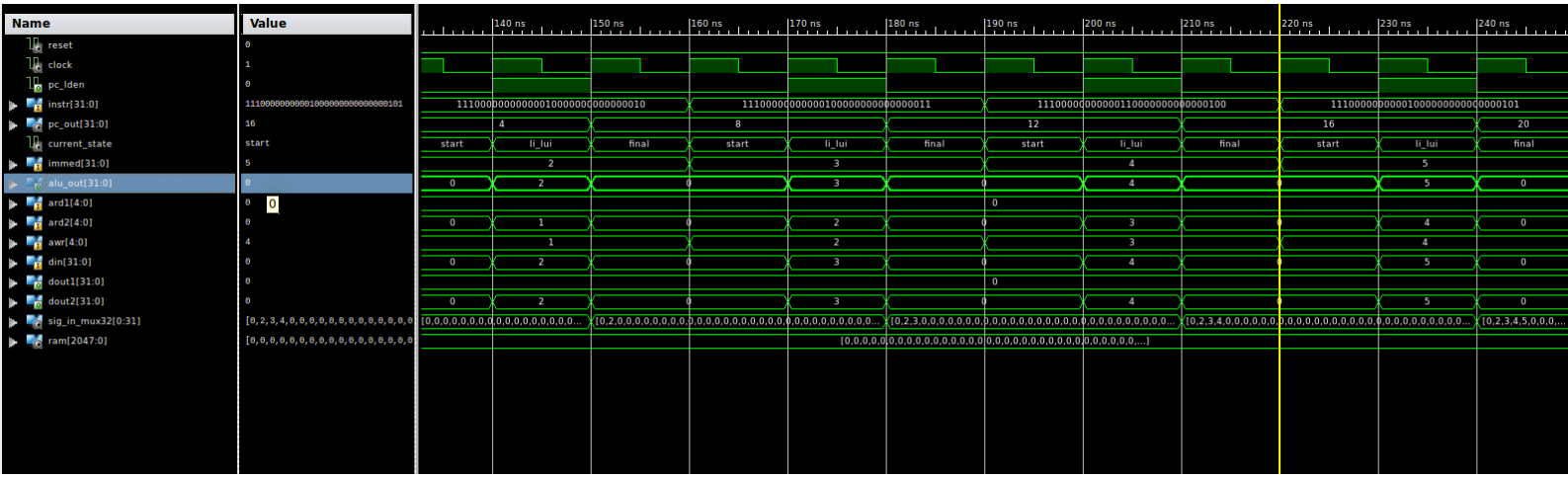
Reset για 10 κύκλους



Nop (add $0, $0, $0)

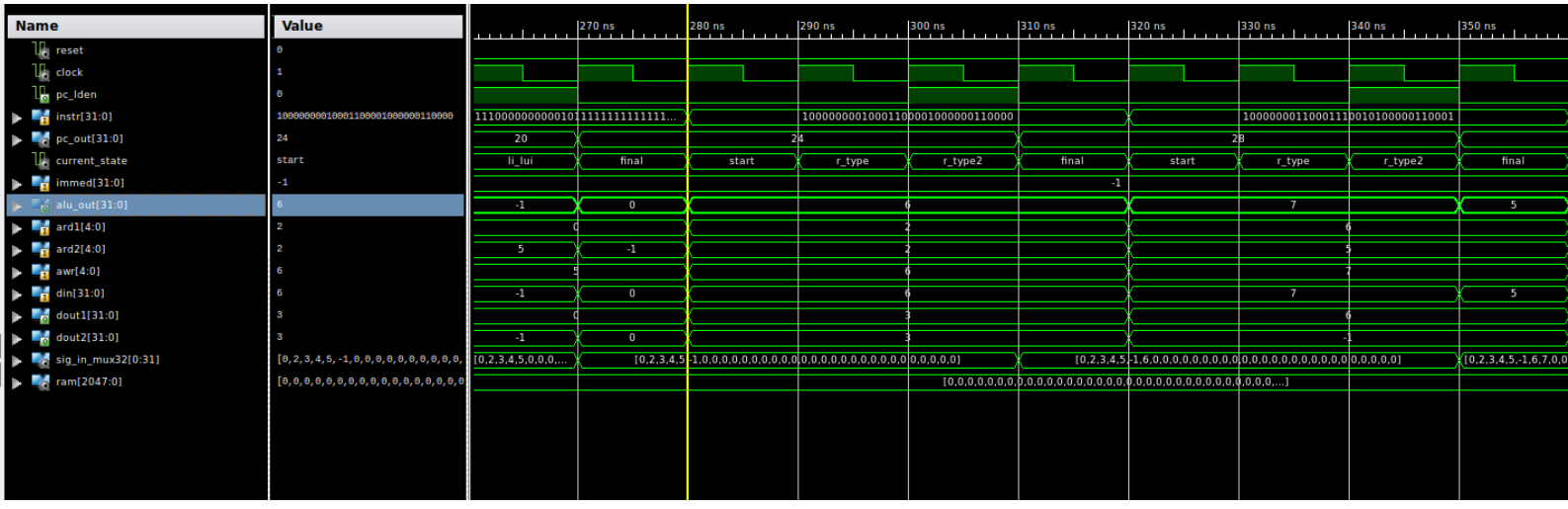


li $1, 2 -- $1 = 2
li $2, 3 -- $2 = 3
li $3, 4 -- $3 = 4
li $4, 5 -- $4 = 5

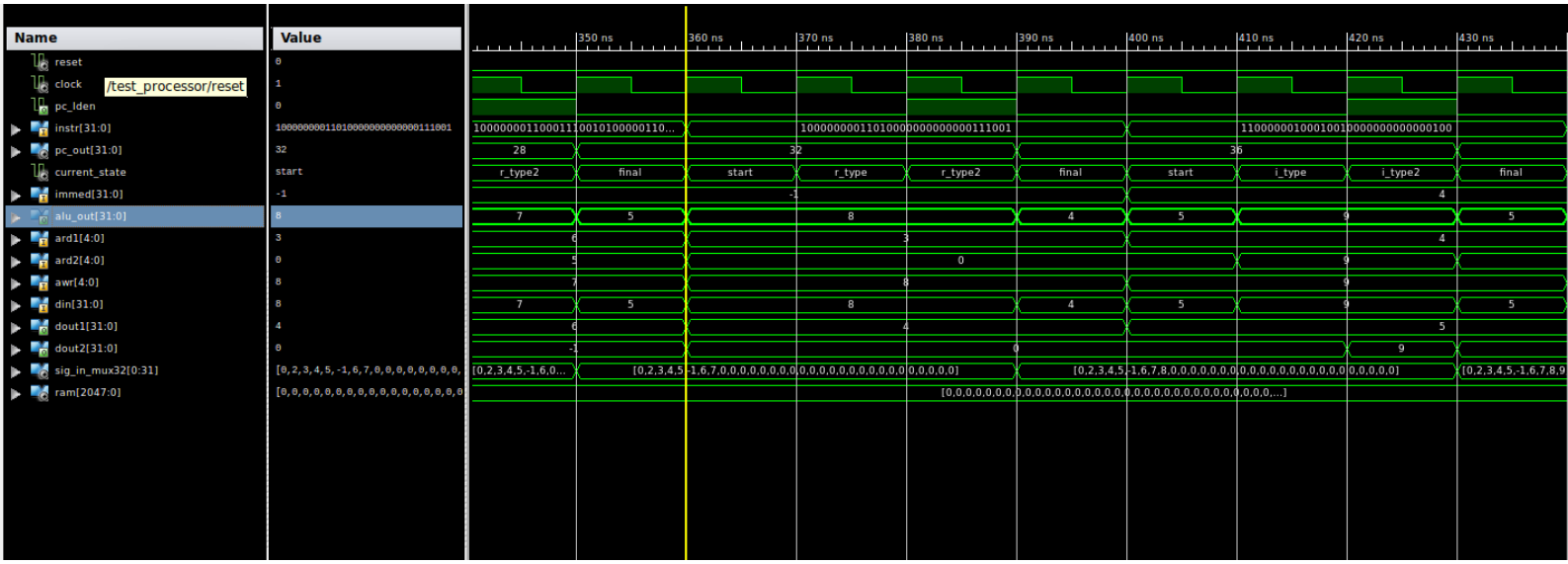


li $5, -1 -- $5 = -1

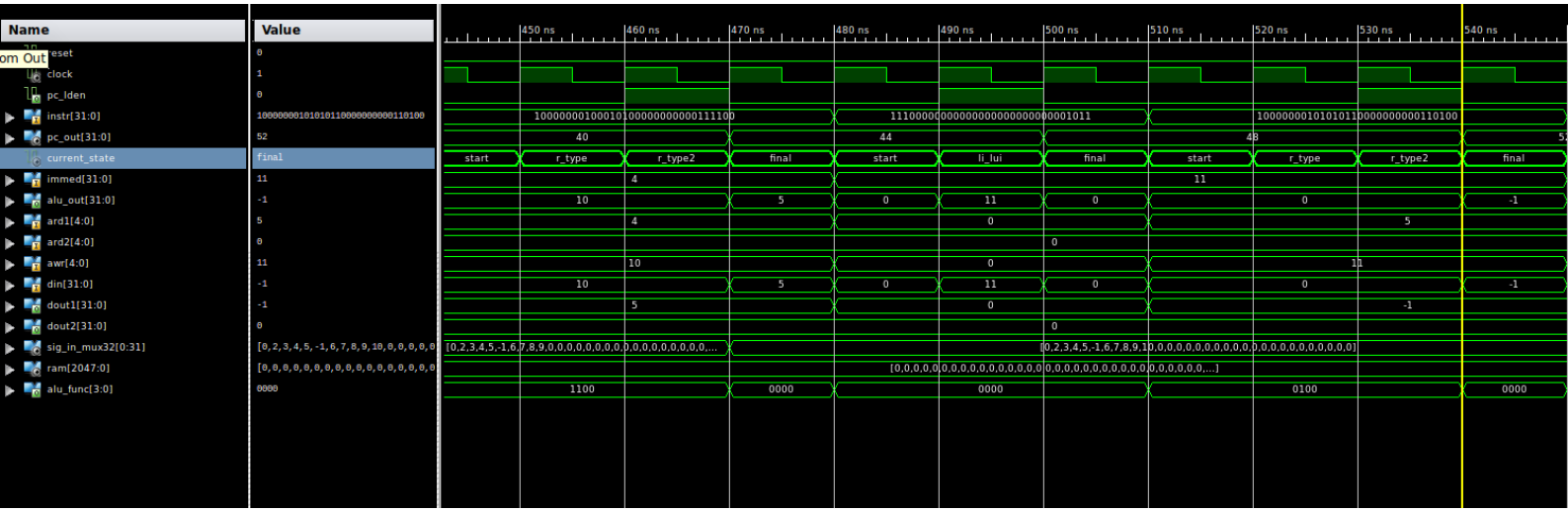
add $6, $2, $2 -- $6 = 6
sub $7, $6, $5 -- $7 = 7



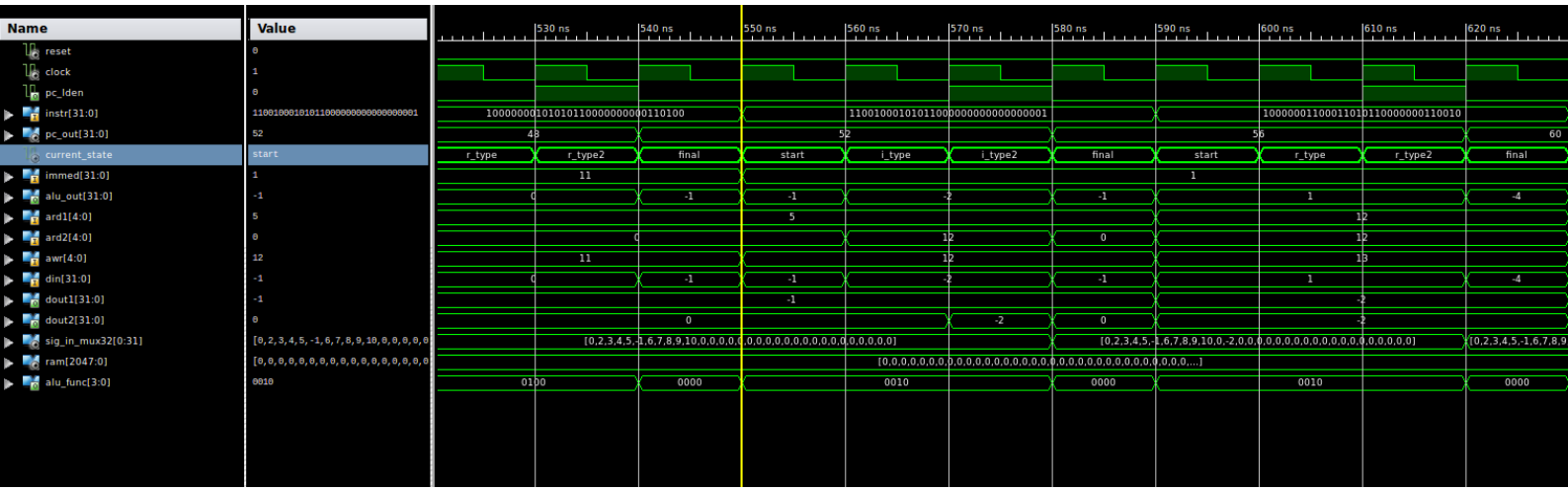
sll $8, $3 -- $8 = 8
addi $9, $4, 4 -- $9 = 9



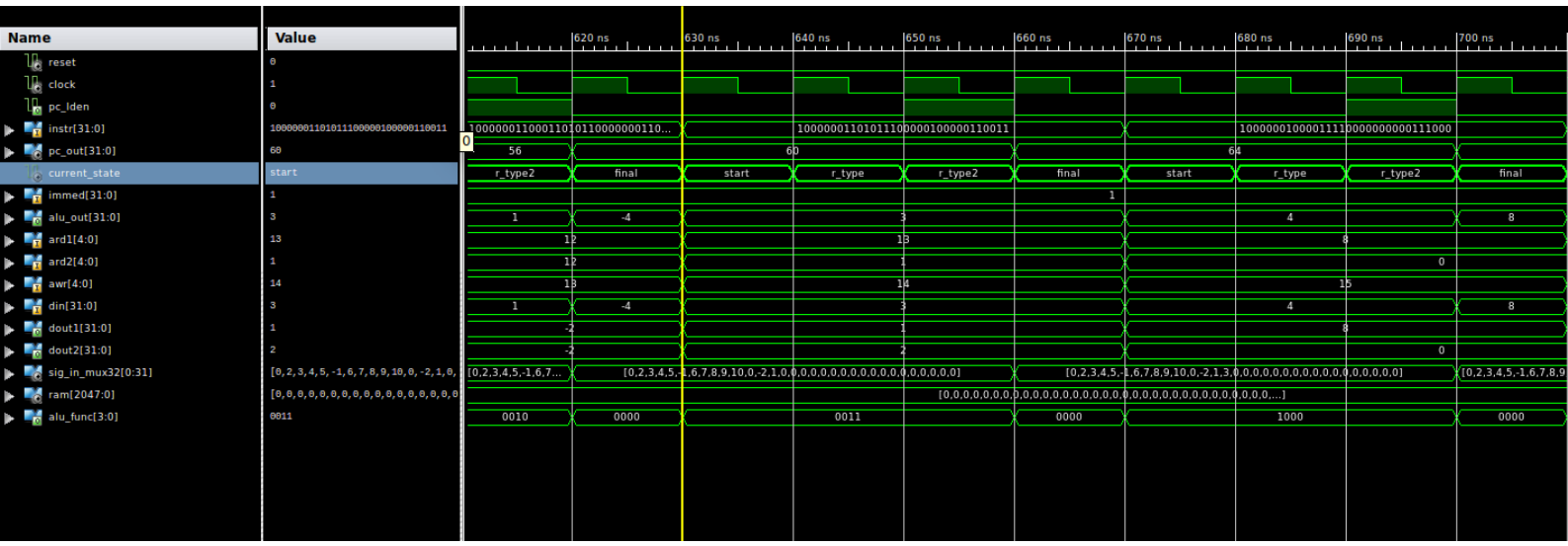
rol $10, $4 -- $10 = 10
li $0, 11 -- nop
not $11, $5 -- $11 = 0



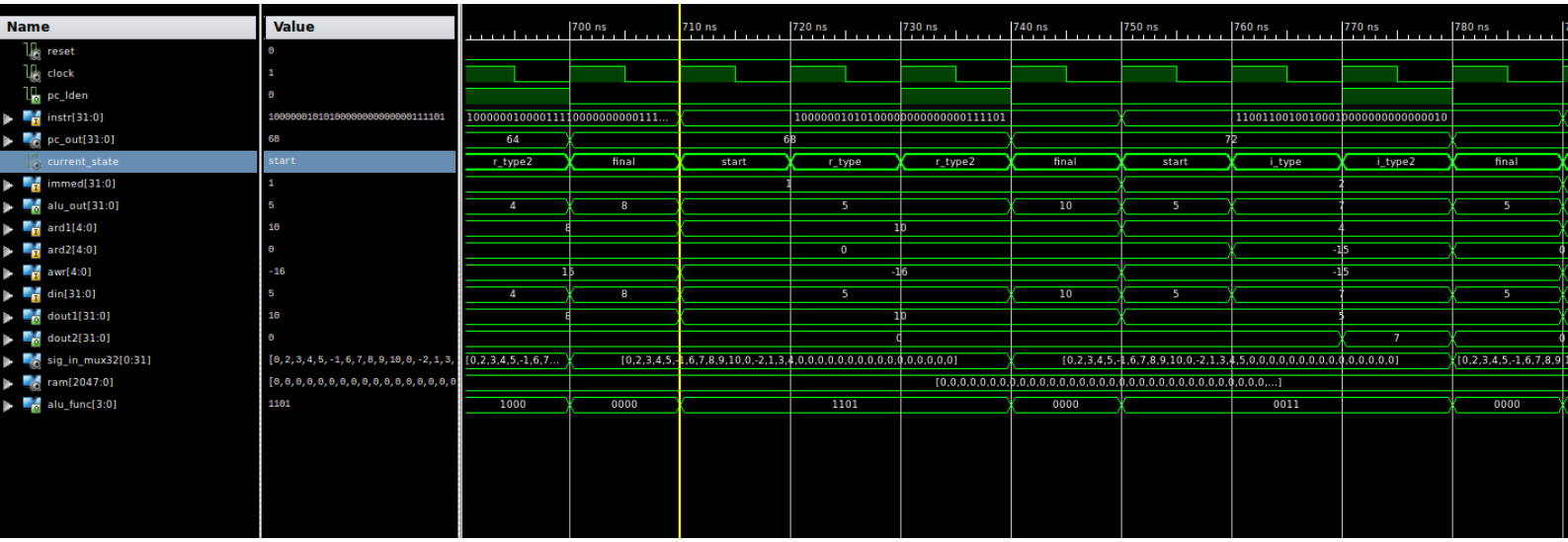
nandi $12, $5, 1 -- $12 = -2
nand $13, $12, $12 -- $13 = 1



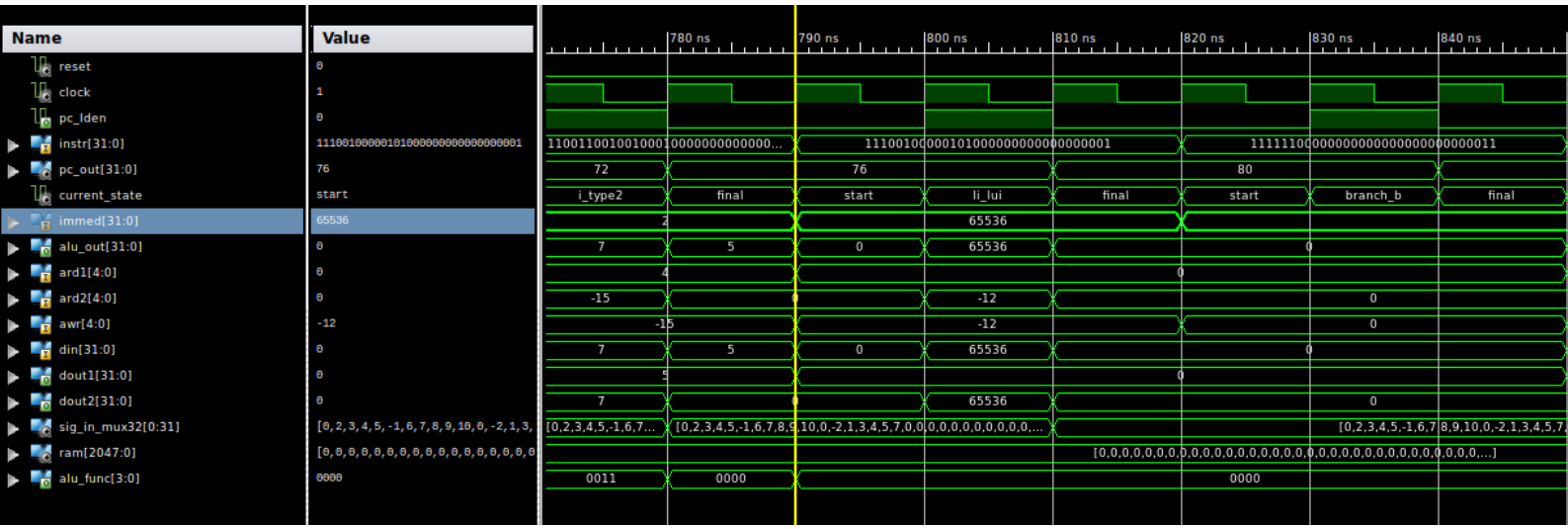
or $14, $13, $1 -- $14 = 3
sra $15, $8 -- $15 = 4



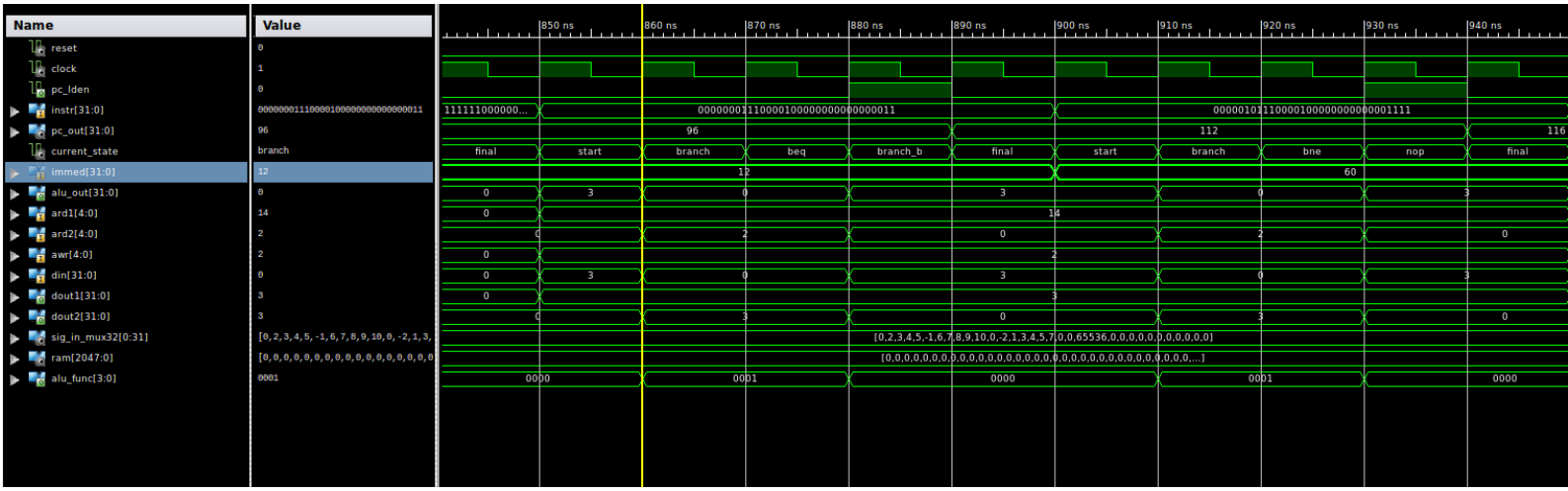
ror $16, $10 -- $16 = 5
ori $17, $4, 2 -- $17 = 7, PC = 72



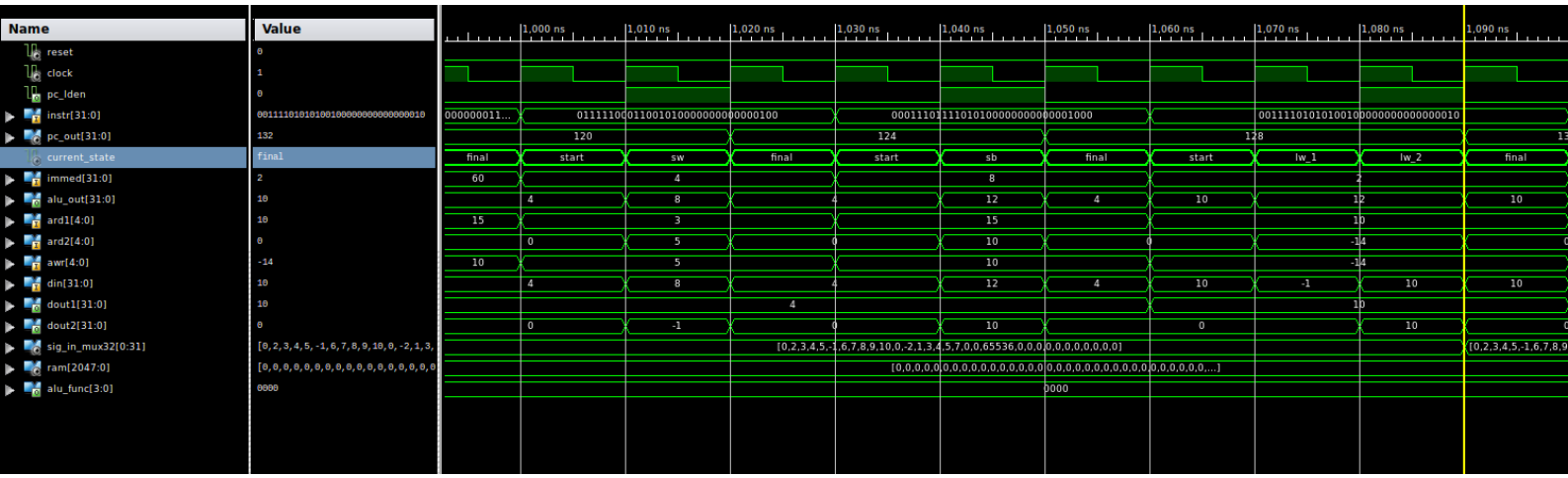
lui $20, 1 -- $20 = 65536, PC = 76
b 3 -- PC = PC + 4 + 12 = 96

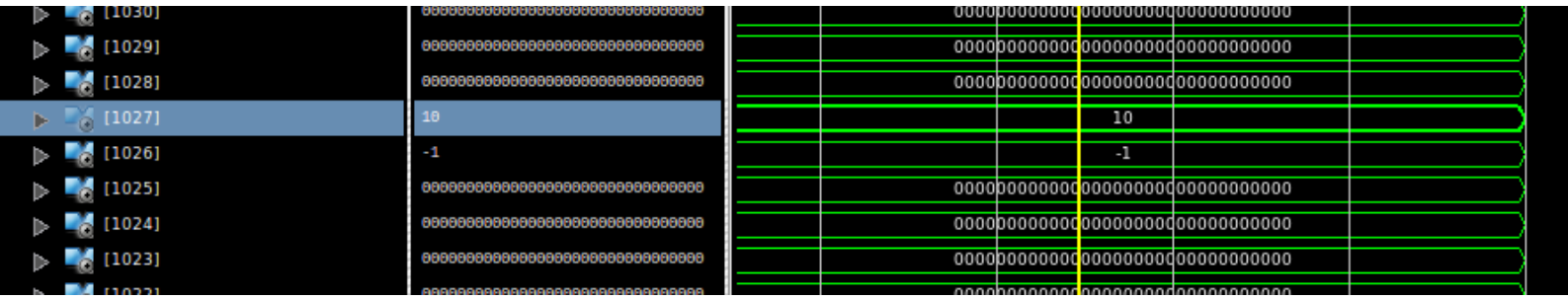


beq $14, $2, 3 -- PC = PC + 4 + 12 = 112
bne $14, $2, 16 -- PC = 116



sw $5, 4($3) -- MEM[1026] = -1
sb $10, 8($15) -- MEM[1027] = 10
lw $18, 2($10) -- $18 = MEM[1027] = 10





lb $19, 8($0) -- $19 = MEM[1026] = 255