ΨΗΦΙΑΚΑ ΣΥΣΤΗΜΑΤΑ HARDWARE

Κόρο Έρικα (9707), Κυριαφίνης Βασίλης (9797)

Contents

1 Modules 2
1.1 Instruction Fetch 2
1.2 Decode Instruction 3
1.3 ALU Stage 4
1.4 Memory stage 4
2 Helper modules: 5
2.1 Multiplexer 5
3 Datapath 6
4 Control 6
4.1 Control signals 6
4.1.1 Register File 6
4.1.2 Instruction decode 7
4.1.3 ALU 7
4.1.4 Memory 7
5 Simulations 8
5.1 Nop instruction 8
5.2 Add instruction 9
5.3 Add immediate instruction 10
5.4 Branch instruction 10
5.5 Branch not equal instruction 11
5.6 Branch equal instruction 11
5.7 Load byte instruction 12
5.8 Load immediate 12
5.9 Load word instruction 13
5.10 No operation instruction 13
5.11 Store byte instruction 14
5.12 Store word instruction 14

1 Modules

1.1 Instruction Fetch

Inputs:

1. [31:0] PC_Immed: Τιμή Immediate για εντολές b, beq, bne.
2. PC_Sel: Επιλογή εισόδου για ενημέρωση του PC (PC+4 ή PC+4+Immed).
3. PC_LdEn: Ενεργοποίηση εγγραφής στον PC.
4. Reset: Είσοδος Reset για αρχικοποίηση του καταχωρητή PC.
5. Clk: Ρολόι.

Outputs:

1. wire [31:0] Instr: Η εντολή που ανακλήθηκε.

Internal signals:

1. reg [31:0] PC: Register που αποθηκεύει την τιμή του Program Counter.
2. wire [2 * 32 - 1:0] mux_input: Bus 64-bits (2 x 32) για την είσοδο δεδομένων στον multiplexer.
3. wire [31:0] mux_to_pc: Η έξοδος του multiplexer για την ανανέωση της τιμής του Program Counter.
4. wire [31:0] pc_to_memory: Bus για την σύνδεση του program counter με την μνήμη.

Η βαθμίδα Instruction Fetch διαβάζει εντολές από την μνήμη που υλοποιείται μέσα στο module IMEM, ανάλογα με την τιμή του program counter (PC). Η τιμή του PC καθορίζεται από το προηγούμενο instruction που εκτελέστηκε, το οποίο καθορίζει το PC_Sel του multiplexer. Αν η προηγούμενη εντολή είναι branch τότε PC_Sel = 1 και ο PC γίνεται PC+4 + Immediate διαφορετικά PC_Sel = 0 και ο PC γίνεται PC+4.

1.2 Decode Instruction

Inputs:

1. [31:0] Instr
2. RF_WrEn: Ενεργοποίηση εγγραφής καταχωρητή.
3. [31:0] ALU_out: Δεδομένα εγγραφής καταχωρητή προερχόμενα από την ALU.
4. [31:0] MEM_out: Δεδομένα εγγραφής καταχωρητή προερχόμενα από τη μνήμη.
5. RF_WrData_sel: Επιλογή πεδίου που καθορίζει την προέλευση δεδομένων προς εγγραφή.
6. RF_Bsel: Επιλογή πεδίου που καθορίζει τον δεύτερο καταχωρητή ανάγνωσης.
7. Clk

Outputs:

1. reg [31:0] Immed: Immediate προς τις επόμενες βαθμίδες.
2. wire [31:0] RF_A: Η τιμή του 1ου καταχωρητή.
3. wire [31:0] RF_B : Η τιμή του 2ου καταχωρητή.

Internal signals:

1. wire [2 * 32 - 1:0] write_select_bus: Συνδέει την έξοδο της ALU και της μνήμης με την είσοδο του πολυπλέκτη για την εγγραφή δεδομένων στην Register File.
2. wire [2 * 5 - 1:0] read_addr_select_bus: Συνδέει τα κομμάτια της εντολής Instr[15:11] και Instr[20:16] στον πολυπλέκτη για την ανάγνωση δεδομένων από την Register File.
3. wire [31:0] write_data: Συνδέει την έξοδο του πολυπλέκτη για την εγγραφή δεδομένων στην Register File.
4. wire [4:0] read_addr_2: Συνδέει την έξοδο του πολυπλέκτη για την ανάγνωση διεύθυνσης στην Register File.

Στην βαθμίδα αυτή υλοποιείται το module Register File και η μετατροπή του Immed από 16-bits σε 32. Στην περίπτωση που το Instruction που δεν είναι τύπου immediate τότε Immed = 0.

1.3 ALU Stage

Inputs:

1. [31:0] RF_A: RF[rs].
2. [31:0] RF_B: RF[rt] ή RF[rd].
3. [31:0] Immed: Immediate.
4. ALU_Bin_sel: Επιλογή Εισόδου B της ALU από RF_B ή Immediate.
5. [3:0] ALU_func: Πράξη ALU.

Outputs:

1. wire [31:0] ALU_out: Αποτέλεσμα ALU.
2. wire zero

Internal signals:

1. wire [2*32 - 1:0] inputs: Είσοδοι του πολυπλέκτη.
2. wire [31:0] mux_to_ALU: Η έξοδος του πολυπλέκτη που πέρναει ως είσοδος στο ALU Module.

Στην βαθμίδα αυτή υλοποιείται το ALU module για την υλοποίηση αριθμητικών και λογικών πράξεων.

1.4 Memory stage

Inputs:

1. clk: Ρολόι
2. Mem_WrEn: Σημαία ενεργοποίησης εγγραφής στη μνήμη.
3. Mem_In_Out_Sel: Σήμα επιλογής για την προσθήκη πολυπλκτών στο module.
4. [31:0] ALU_MEM_Addr: Αποτέλεσμα ALU (βλέπε εντολές lb, sb, lw, sw).
5. [31:0] MEM_DataIn: Αποτέλεσμα RF[rd] για αποθήκευση στη μνήμη για εντολές swap και sb, sw.

Outputs:

1. [31:0] MEM_DataOut: Δεδομένα που φορτώθηκαν από τη μνήμη προς εγγραφή σε καταχωρητή για εντολές lb, lw.

Internal signals:

1. wire [32 * 2 - 1:0] input_mux_data: Είσοδος στον πολυπλέκτη στην είσοδο της μνήμης.
2. wire [31:0] input_mux_to_mem: Συνδέει την έξοδο του πολυπλέκτη με την είσοδο στην μνήμη.

3. wire [32 * 2 - 1:0] output_mux_data: Είσοδος στον πολυπλέκτη που βρίσκεται στην έξοδο της μνήμης.
4. wire [31:0] mem_to_output_mux: Συνδέει την έξοδο της μνήμης με τις είσοδους του πολυπλέκτη που βρίσκεται στην έξοδο της μνήμης.

Στην βαθμίδα αυτή υλοποιείται η μνήμη και προστέθηκαν δύο multiplexers για να πραγματοποιηθούν οι εντολές sb και lb. Τα δεδόμενα που πρέπει να αποθηκευτούν έρχονται σε 32-bit μορφή, ωστόσο το τελευταίο byte περιέχει την επιθυμητή πληροφορία και όλα τα υπόλοιπα στοιχεία πρέπει να είναι 0. Για να επιτευχθεί αυτό γίνεται ένα bitwise and μεταξύ του δεδομένου αριθμού και του 00000000_00000000_00000000_11111111 αφού x and 1 = x, δηλαδή υλοποιείται masking του αριθμού.

2 Helper modules:

2.1 Multiplexer

Parameters:

1. BUS_WIDTH: Τα bits από μια είσοδο.
2. SEL: Ο αριθμός των select bits.

Inputs:

1. [(BUS_WIDTH*(2**SEL))-1:0] Din: Οι είσοδοι στον multiplexer.
2. [SEL-1:0] Sel: Η τιμή του SEL που καθορίζει την έξοδο του πολυπλέκτη.

Outputs:

1. reg [BUS_WIDTH-1:0] Dout: Η έξοδος του multiplexer.

Internal signals:

1. reg [BUS_WIDTH - 1:0] inputs [2 ** SEL - 1:0]: Ένα array από 2^SEL registers μήκους BUS_WIDTH bits που κρατάει την κάθε είσοδο των 32 bits.

Η είσοδος Din που δίνουμε στον multiplexer είναι ένα bus που είναι 2^SEL x BUS_WIDTH bits, σε τμήματα του οποίου βρίσκονται όλες οι είσοδοι που δίνονται στον multiplexer.

3 Datapath

Στο Instruction Fetch διαβάζεται η επόμενη εντολή από την μνήμη που πρέπει να εκτελεστεί και περνιέται στο στάδιο του decode. Από τμήματα του instruction εξάγονται οι διευθύνσεις μνήμης για την Register File , ώστε να βγουν τα περιεχόμενα των καταχωρητών ως σήματα RF_A και RF_B. Ανάλογα το instruction, καθορίζεται και το Immediate. Στη συνέχεια, το RF_A περνάει στην πρώτη είσοδο της ALU και το RF_B με το Immediate περνάνε ως είσοδοι στον πολυπλέκτη, η έξοδος του οποίου καθορίζει ποια θα είναι η δεύτερη είσοδος στην ALU. Έχοντας ολοκληρώσει την λειτουργία της, η έξοδός της, όταν πρόκειται για memory instructions περνάει ως είσοδος στην μνήμη και καθορίζει σε ποια διεύθυνση θα πραγματοποιηθεί εγγραφή ή ανάγνωση διαφορετικά, περνάει στο decode stage. Επίσης, ως είσοδος στο memory stage περνάει και το περιεχόμενο του καταχωρητή RF_B, που χρησιμοποιείται ως δεδομένα εισόδου στην περίπτωση εντολής αποθήκευσης. Αυτό μαζί με το σήμα (RF_B & 00000000_00000000_00000000_11111111) εισάγονται στην μνήμη μέσω ενός πολυπλέκτη. Η έξοδος της μνήμης MEM_Out και (MEM_Out & 00000000_00000000_00000000_11111111) περνάνε σ' έναν πολυπλέκτη και η έξοδός του δίνεται ως είσοδος σ' έναν πολυπλέκτη που καθορίζει τα δεδομένα εγγραφής στην Register File του decode stage μαζί με την έξοδο που είχε βγάλει η ALU.

4 Control

Σκοπός του Control module είναι να ενεργοποιεί τα κατάλληλα σήματα ελέγχου για την ορθή εκτέλεση των εντολών. Για τον σκοπό αυτό δημιουργήθηκαν στάδια, σε καθένα από τα οποία ενεργοποιούνται διαφορετικά σήματα για την εξέλιξη της εντολής όπως αυτή πραγματοποιείται. Για την διευκόλυνση της διεργασίας χρησιμοποιήθηκε ένας καταχωρητής **adder**, που αυξάνεται στην θετική ακμή του ρολογιού. Αυτός ο καταχωρητής καθορίζει και σε ποια φάση από την εκτέλεση της εντολής βρισκόμαστε ώστε να διεγερθούν τα κατάλληλα σήματα. Για κάθε εντολή υλοποιείται μια ξεχωριστή περίπτωση σε μια **else**, **if else**, **else** δομή, ενώ εντολές οι οποίες έχουν παρόμοια εκτέλεση, όπως οι εντολές της ALU, ομαδοποιούνται σε μία περίπτωση.

4.1 Control signals

4.1.1 Register File

1. PC_Sel: Η τιμή του εξαρτάται από τον τύπο εντολής που εκτελείται. Αν η εντολή είναι τύπου branch, παίρνει τιμή 1 αλλιώς 0.

2. PC_LdEn: Γίνεται 1 στο τέλος της εκτέλεσης της κάθε εντολής για να αλλάξει η τιμή του program counter και να προχωρήσει η εκτέλεση του προγράμματος.
3. Reset: Παίρνει την τιμή 1 όταν θέλουμε να αρχίσει από την αρχή η εκτέλεση του προγράμματος.

4.1.2 Instruction decode

1. RF_B_sel: Ενεργοποιείται για immediate εντολές.
2. RF_WrData_sel: Παίρνει την τιμή 1 μόνο όταν εκτελείται εντολή η οποία γράφει από την μνήμη στην Register File.
3. RF_WEn: Γίνεται 1 όταν πρέπει να γραφτεί το αποτέλεσμα της μνήμης η της ALU σε έναν καταχωρητή.

4.1.3 ALU

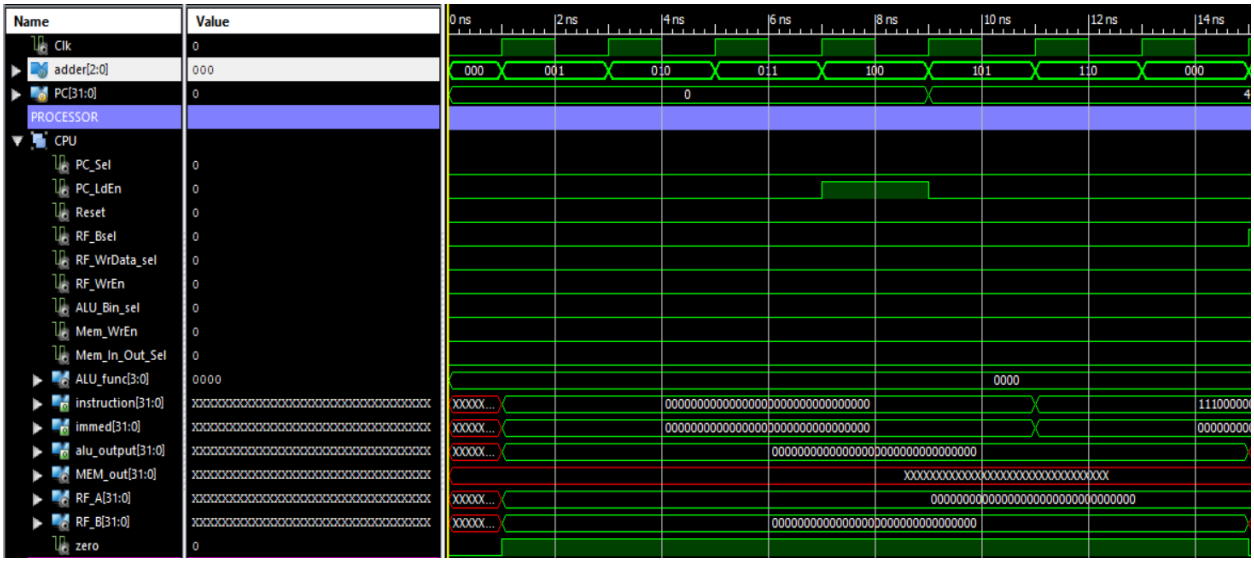
1. ALU_Bin_sel: Ενεργοποιείται όταν πρέπει να περάσει το immediate σαν δεύτερη είσοδος στην ALU.
2. ALU_func: Χρησιμοποιείται κυρίως στις αριθμητικές εντολές και σε κάποιες άλλες περιπτώσεις (andi, ori, bne, etc).

4.1.4 Memory

1. MEM_WrEn: Ενεργοποιείται για τις εντολές store (sb, sw).
2. Mem_In_Out_Sel: Γίνεται 1 για τις εντολές byte (sb, lb).

5 Simulations

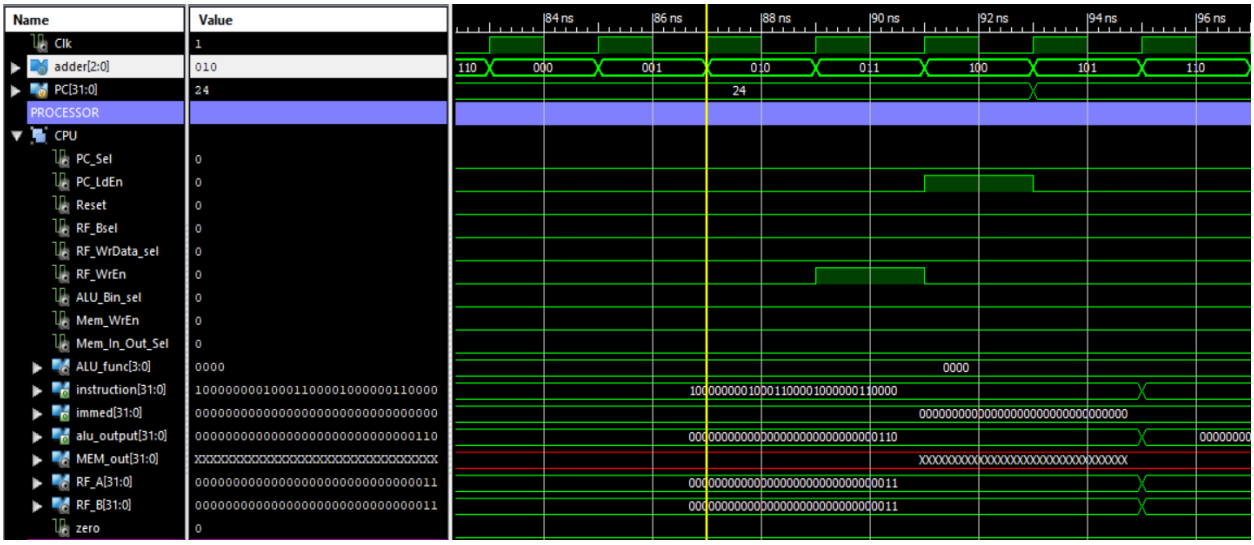
5.1 Nop instruction



Αυτή είναι και η αρχή της προσομοίωσης. Στην παραπάνω εικόνα φαίνεται ο μηχανισμός με τον οποίο χωρίζεται το κάθε instruction σε επιμέρους φάσεις. Ο register adder ξεκινάει από το 0 και αυξάνεται σε κάθε κύκλο κατά 1. Στο συγκεκριμένο instruction στον 5ο κύκλο (adder = 100_2) το PC_LdEn γίνεται 1. Για να διαβαστεί η επόμενη εντολή από την μνήμη χρειάζεται 1 κύκλο. Κάθε εντολή αυξάνει τον PC και έπειτα τερματίζει. Ο adder register φτάνει μέχρι την τιμή 110_2 αυτό γίνεται γιατί η εντολή που παίρνει τους περισσότερους κύκλους χρειάζεται 5 κύκλους για να εκτελεστεί συν έναν κύκλο για να διαβαστεί το επόμενο instruction. Όσες εντολές χρειάζονται λιγότερους κύκλους απλά αναμένουν μέχρι να γίνει ο adder reset και να ξεκινήσει η επόμενη εντολή.

Επίσης σε πολλά σημεία κάποια σήματα, κυρίως από την μνήμη ή την register file, φαίνεται ότι είναι απροσδιόριστα. Αυτό συμβαίνει γιατί οι εντολές που εκτελούνται εκείνη την στιγμή διαβάζουν από register ή μνήμη που εκείνη την στιγμή δεν έχει αρχικοποιηθεί. Αυτό δεν αποτελεί πρόβλημα γιατί οι συγκεκριμένες εντολές δεν κάνουν χρήση αυτών των τιμών.

5.2 Add instruction

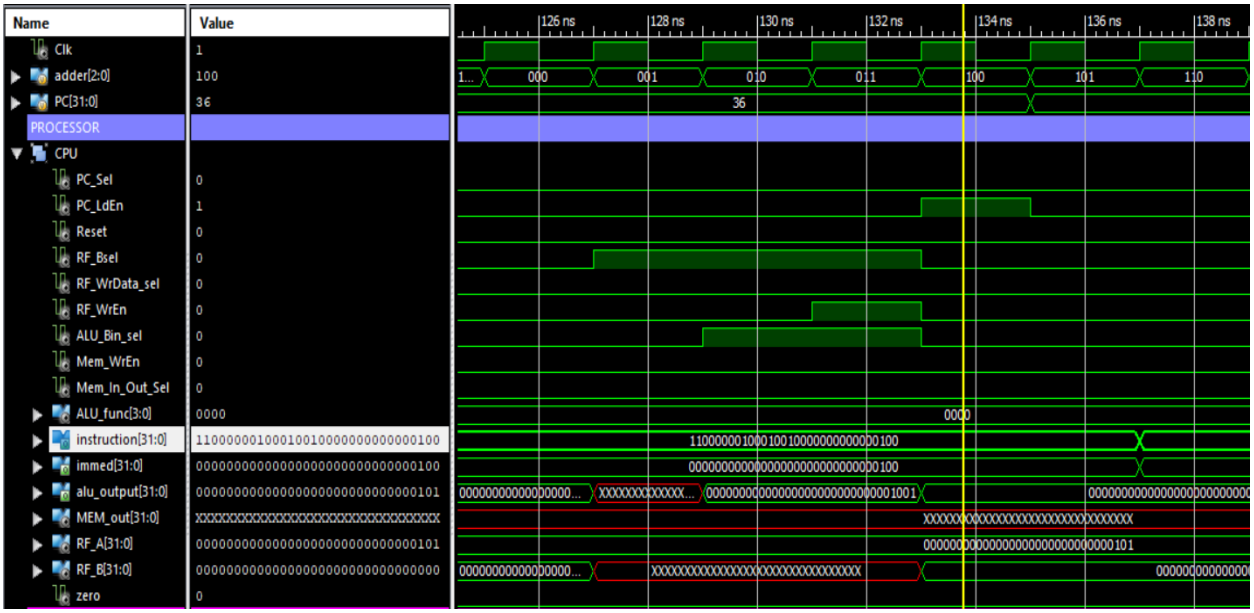


Απ' αυτή την προσομοίωση παρατηρείται ότι στην δεύτερη φάση (adder = 010_2) τα σήματα ALU_Bin_Sel = 0 και το ALU_func = 0000. Στην επομένη φάση με adder = 011_2 το RF_Wen = 1 ώστε να καταγραφεί στο register file του decode stage το alu_output, γι' αυτό εξάλλου και το RF_Write_Data = 0 στον πολυπλέκτη για να επιλεχθεί το alu_output στην έξοδο. Τέλος στην τέταρτη φάση (adder = 100_2), ενεργοποιείται το PC_LdEn = 1 για να ξεκινήσει η διαδικασία ανάγνωσης της επόμενης εντολής από την μνήμη.

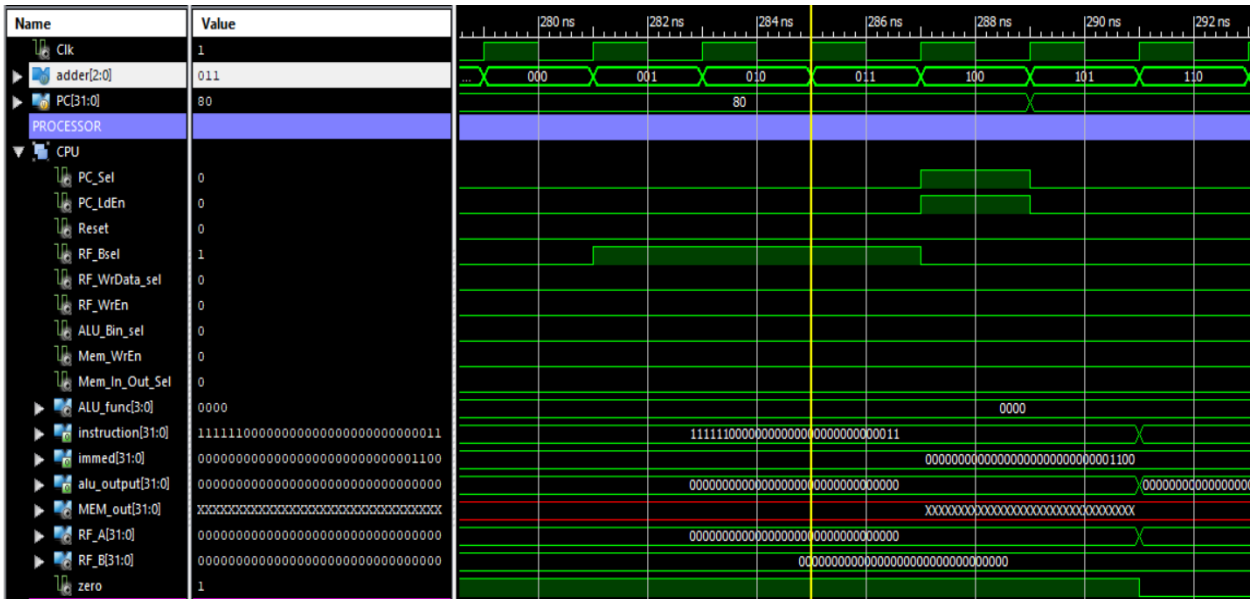
Η add εντολή είναι αντιπροσωπευτική για όλες τις αριθμητικές και λογικές εντολές που εκτελούνται στην ALU. Η μόνη διαφορά είναι ότι αλλάζει το ALU_func για την κάθε εντολή. Για τον λόγο αυτό δεν παρουσιάζονται οι υπόλοιπες εντολές αυτού του τύπου.

Η ίδια λογική ακολουθείται και παρακάτω. Όσες εντολές είναι παρόμοιες ομαδοποιούνται και εκπροσωπούνται από μία εντολή.

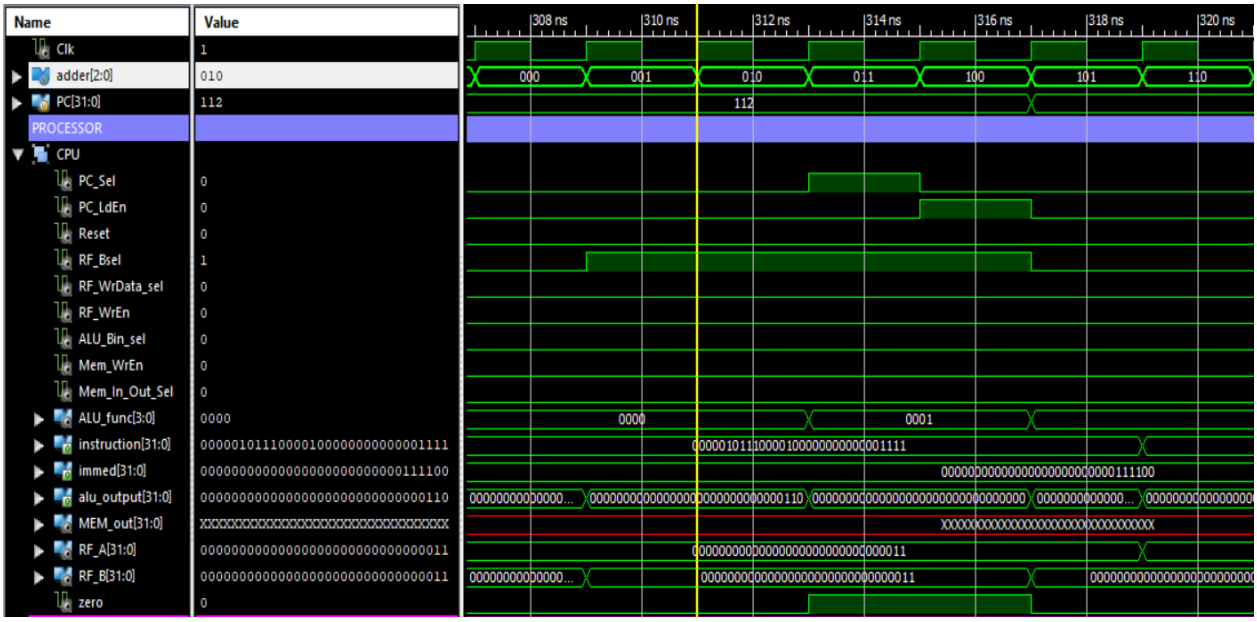
5.3 Add immediate instruction



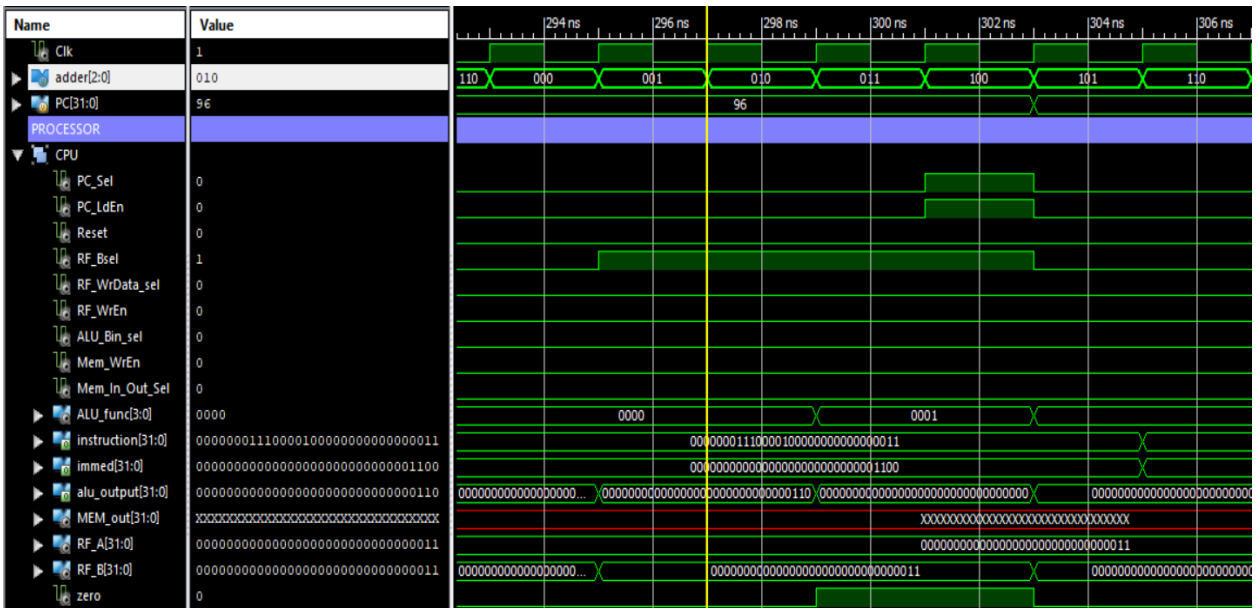
5.4 Branch instruction



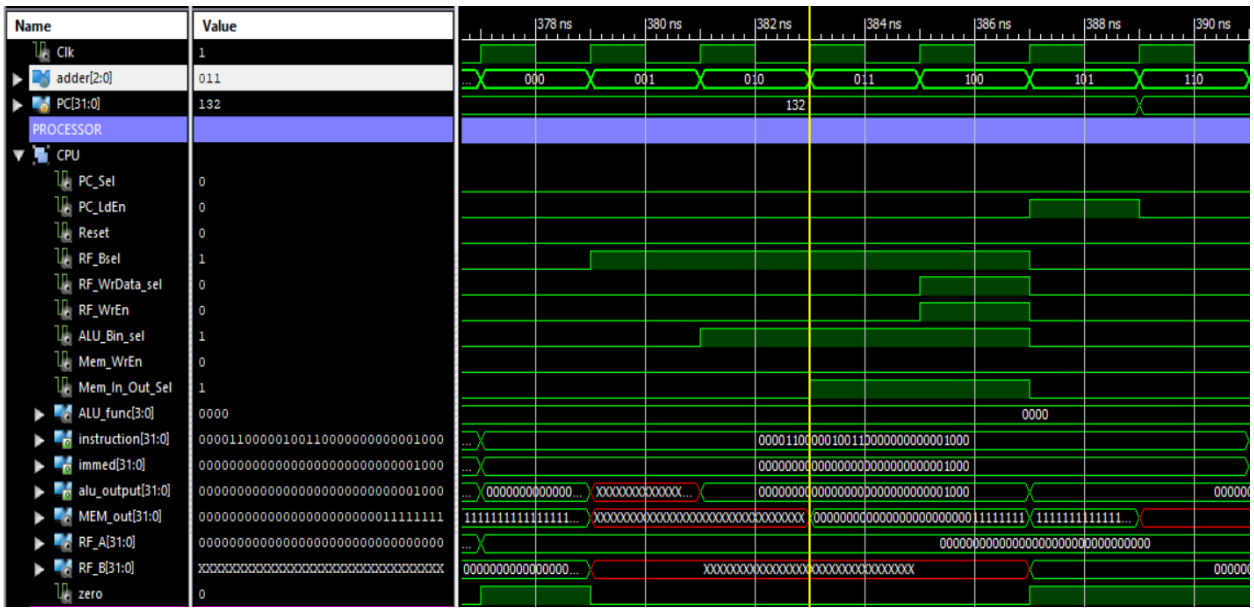
5.5 Branch not equal instruction



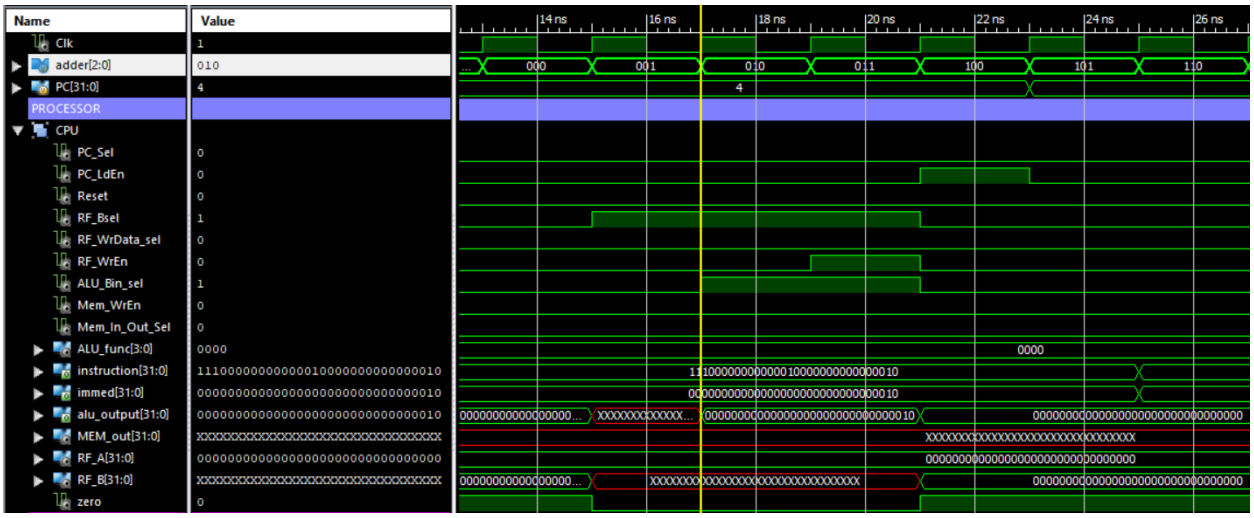
5.6 Branch equal instruction



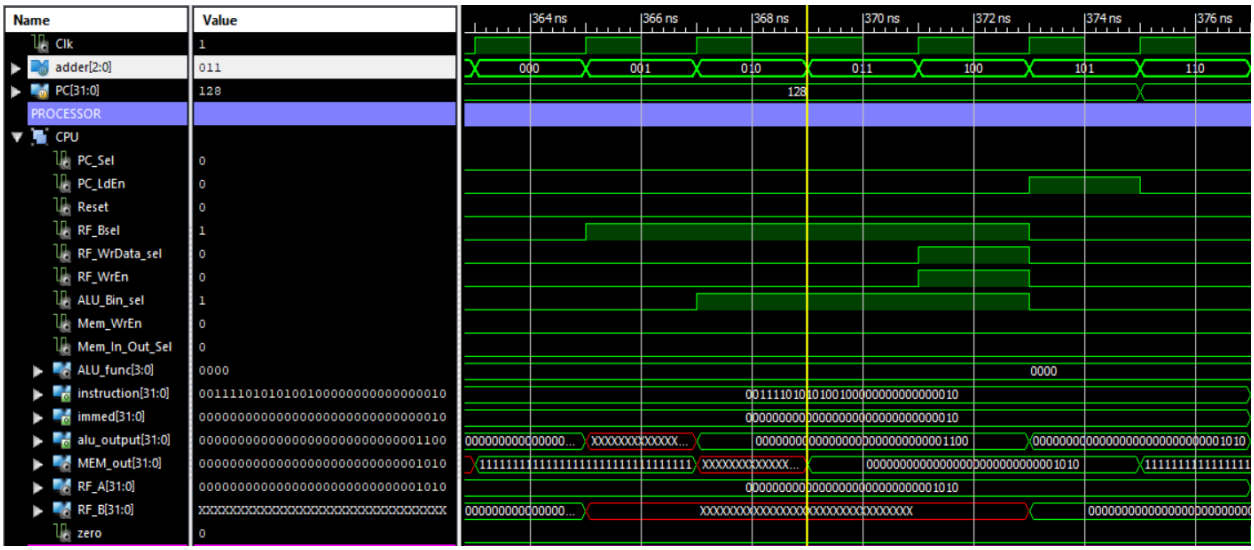
5.7 Load byte instruction



5.8 Load immediate

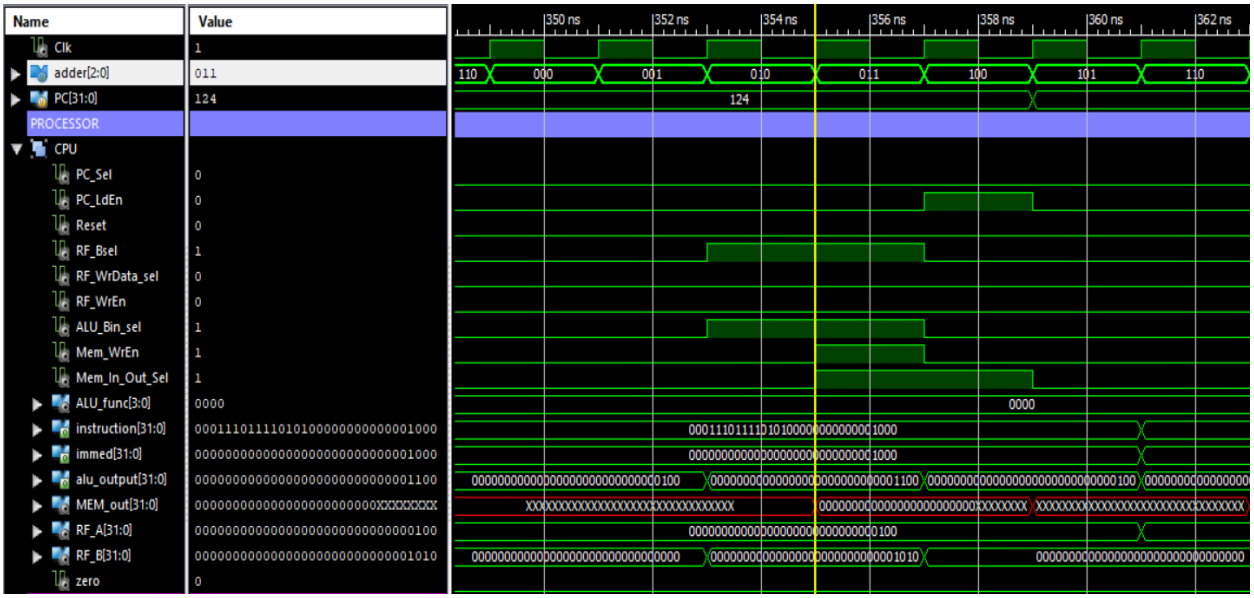


5.9 Load word instruction



5.10 No operation instruction

5.11 Store byte instruction



5.12 Store word instruction