15/3/2017

Αναφορά Εργαστηρίου 3

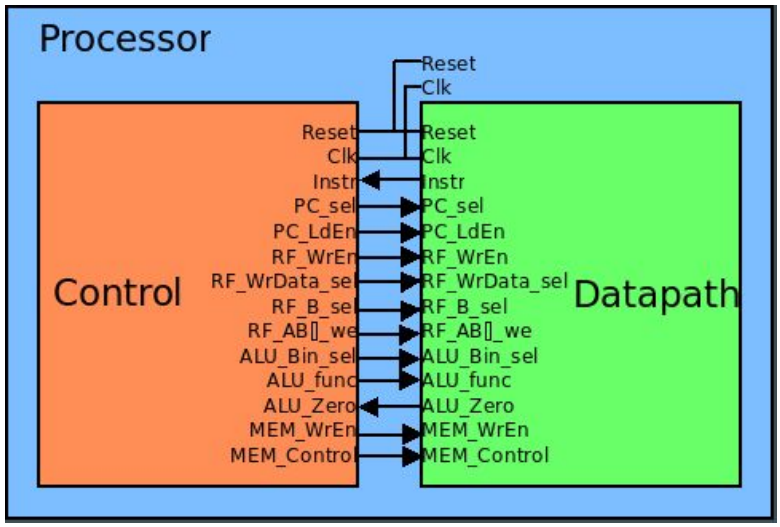
Οργάνωση Υπολογιστών

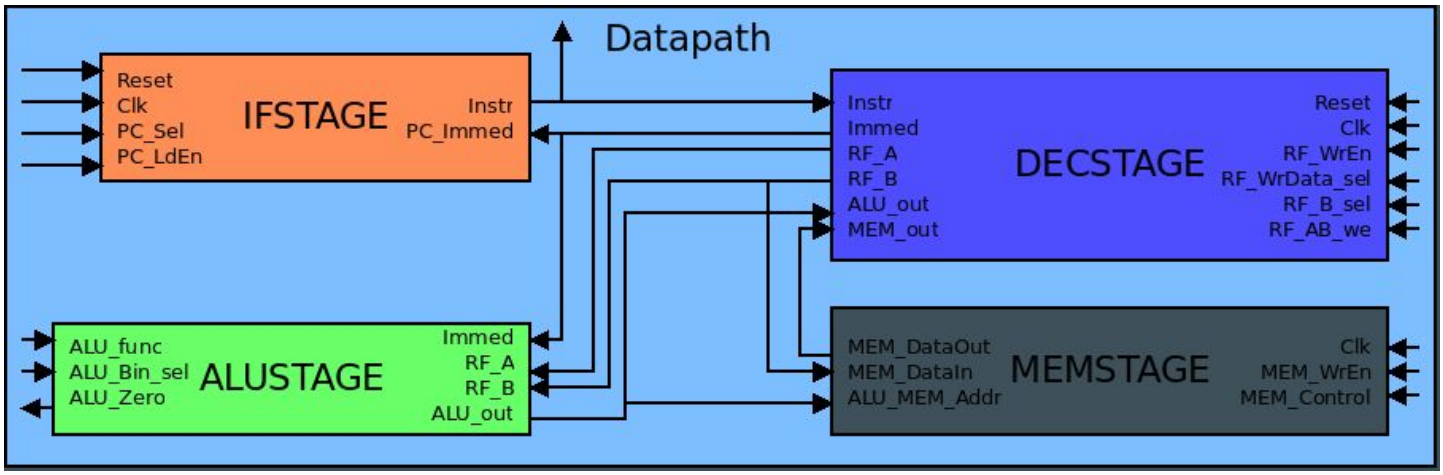
Ομάδα Εργασίας: Κριθαράκης Εμμανουήλ, Φωτάκης Τζανής

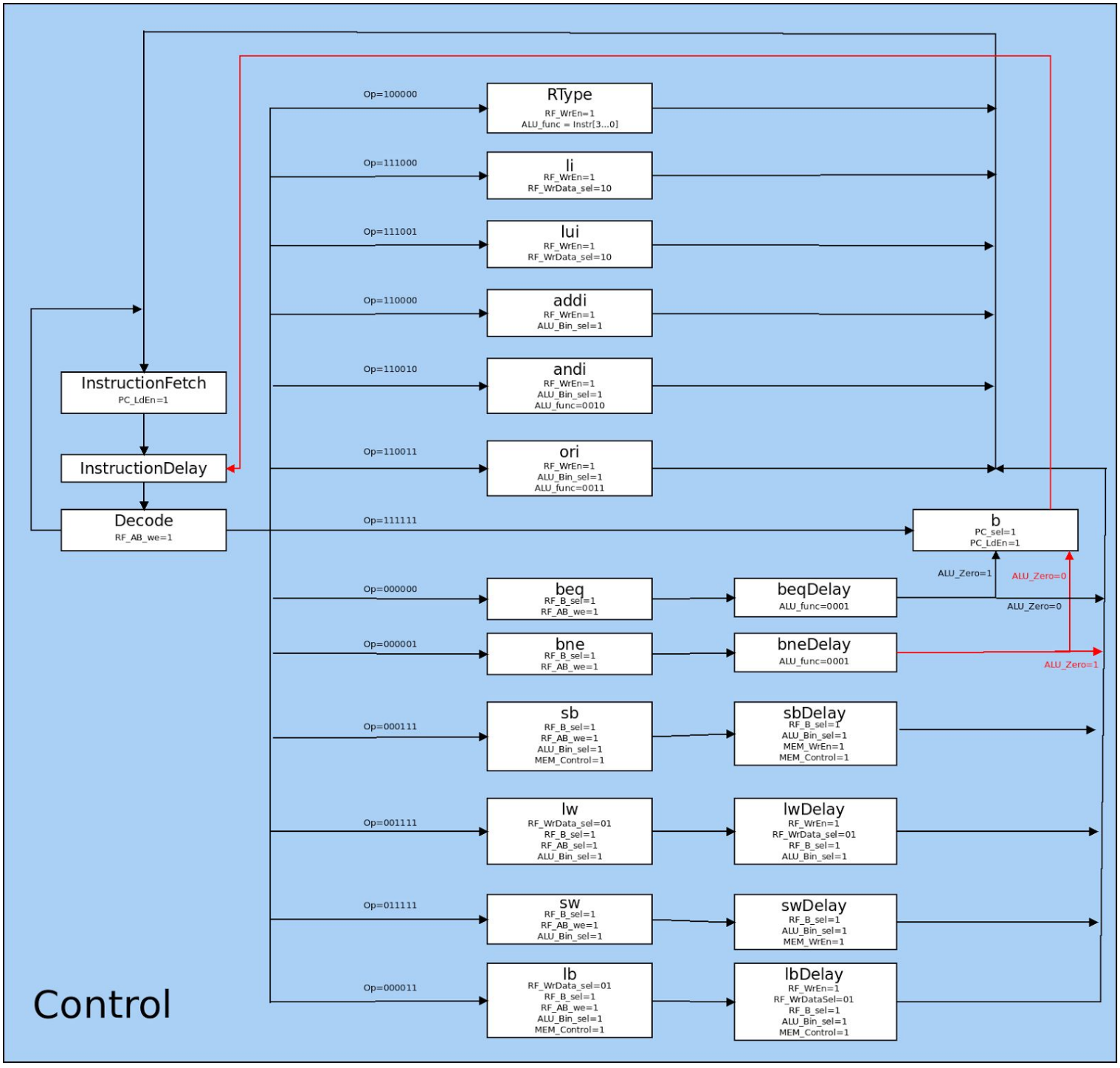
Κωδικός Ομάδας: LAB31231483

Προεργασία

Ως προεργασία του συγκεκριμένου εργαστηρίου είχε ζητηθεί όπως είναι φυσικό ο κώδικας των διαφόρων modules που περιγράφονται παρακάτω αλλά και σχεδιαγράμματα που παρουσιάζουν την συνδεσμολογία μεταξύ των διαφόρων ζητούμενων modules που κληθήκαμε να αναπτύξουμε. Παρακάτω παρατίθενται τα σχετικά σχεδιαγράμματα.






InstructionFetch
PC_LdEn=1
InstructionDelay
Decode
RF_AB_we=1
Op=100000
RType
RF_WrEn=1
ALU_func = Instr[3...0]
Op=111000
Ii
RF_WrEn=1
RF_WrData_sel=10
Op=111001
lui
RF_WrEn=1
RF_WrData_sel=10
Op=110000
addi
RF_WrEn=1
ALU_Bin_sel=1
Op=110010
andi
RF_WrEn=1
ALU_Bin_sel=1
ALU_func=0010
Op=110011
ori
RF_WrEn=1
ALU_Bin_sel=1
ALU_func=0011
Op=111111
b
PC_sel=1
PC_LdEn=1
ALU_Zero=1
ALU_Zero=0
ALU_Zero=0
Op=000000
beq
RF_B_sel=1
RF_AB_we=1
beqDelay
ALU_func=0001
Op=000001
bne
RF_B_sel=1
RF_AB_we=1
bneDelay
ALU_func=0001
ALU_Zero=1
Op=000111
sb
RF_B_sel=1
RF_AB_we=1
ALU_Bin_sel=1
MEM_Control=1
sbDelay
RF_B_sel=1
ALU_Bin_sel=1
MEM_WrEn=1
MEM_Control=1
Op=001111
lw
RF_WrData_sel=01
RF_B_sel=1
RF_AB_sel=1
ALU_Bin_sel=1
lwDelay
RF_WrEn=1
RF_WrData_sel=01
RF_B_sel=1
ALU_Bin_sel=1
Op=011111
sw
RF_B_sel=1
RF_AB_we=1
ALU_Bin_sel=1
swDelay
RF_B_sel=1
ALU_Bin_sel=1
MEM_WrEn=1
Op=000011
lb
RF_WrData_sel=01
RF_B_sel=1
RF_AB_we=1
ALU_Bin_sel=1
MEM_Control=1
lbDelay
RF_WrEn=1
RF_WrDataSel=01
RF_B_sel=1
ALU_Bin_sel=1
MEM_Control=1
Control

Περιγραφή της Άσκησης

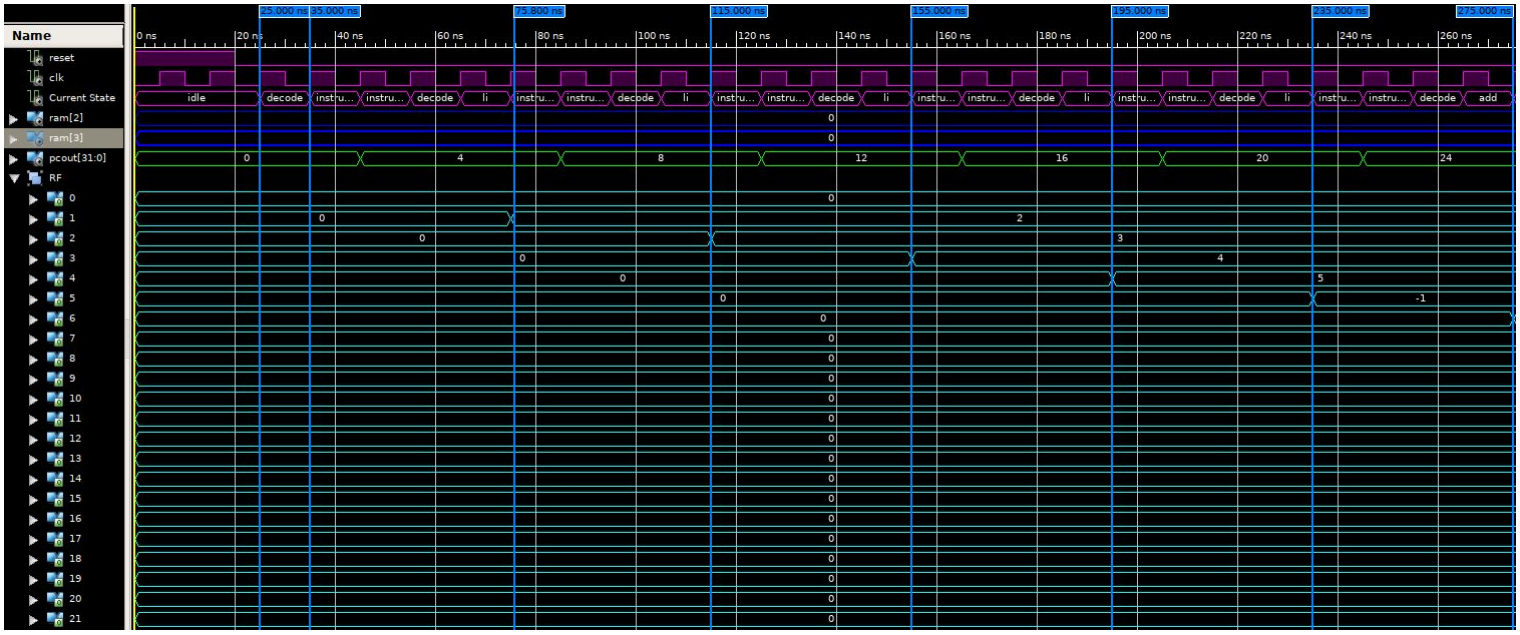
Σκοπός του εργαστηρίου ήταν να ενωθούν τα τμήματα του Datapath του προηγούμενου εργαστηρίου και να σχεδιαστεί κατάλληλα το control έτσι ώστε να εκτελούνται ορθά οι εντολές του instruction set CHARIS-4. Παρακάτω αναλύονται οι καταστάσεις της fsm για κάθε εντολή του instruction set:

- *R-type εντολές*:Σε κάθε τέτοιου τύπου εντολή η fsm ξεκινάει από την κατάσταση InstructionFetch κατα την οποία στον καταχωρητή PC εγγράφεται (PC_ldEn=1)το αποτέλεσμα της πράξης PC=PC+4 (PC_sel=0).Στην συνέχεια η επόμενη κατάσταση (InstructionDelay) είναι υπεύθυνη να παρατείνει τον χρόνο εκτέλεσης της R-type εντολής κατά ένα κύκλο ρολογιού με σκοπό να “έλθει” από την μνήμη εντολών η επιθυμητή εντολή. Εφόσον έρθει τότε περνάει στην κατάσταση Decode. Στην κατάσταση αυτή, ”αποκωδικοποιούμε” την εντολή λαμβάνοντας τα δεδομένα για τους καταχωρητές R[rt], R[rs] για την κάθε R-type εντολή και τα τοποθετούμε στους κατάλληλους registers της Register-file. Επιπλέον οφείλουμε να αναφέρουμε πως για την ορθή λειτουργία του processor προσθέσαμε 2 καταχωρητές Rf[A] και Rf[B] στην έξοδο της Register-file τους οποίους εγγράφουμε μόνο κατά το στάδιο του Decode. Φυσικά για να γίνει αντιληπτό για ποία εντολή γίνεται λόγος ελέγχουμε τα 6 msb bits της εντολής(opcode) και μιας και είναι R-type τα 6 lsb bits της (func). Με τα opcode και func η fsm κατανοεί ποία εντολή θέλουμε να εκτελέστεί και προσδιορίζει κατάλληλα το control unit της ALU. Ως είσοδοι στην ALU εισάγονται οι καταχωρητές Rf[A] και Rf[B](οι οποίοι έχουμε ενημερωθεί από την έξοδο της Register-file) με κατάλληλη ανάθεση του control bit ενός πολυπλέκτη για την επιλογή του Rf[B] (Συγκεκριμένα ALU_bin_sel=0). Το αποτέλεσμα της εγγράφεται στην Register file στον καταχωρητή R[rd] με κατάλληλη ανάθεση του control bit ενός πολυπλέκτη για την επιλογή ALU_out.
- *I-type εντολές:* Τέτοιου είδους εντολές διαφοροποιούνται σε ελάχιστα σημεία απο τις R-type εντολές. Συγκεκριμένα η πρώτη διαφοροποίηση στην αλληλουχία της fsm έγκειται στο στάδιο Decode, όπου και στην αποκωδικοποίηση λαμβάνονται τα δεδομένα για τον καταχωρητή R[rd] και το Immediate. Στην συνέχεια μόνο με την γνώση του opcode η fsm κατανοεί ποία εντολή θέλουμε να εκτελέστεί και προσδιορίζει κατάλληλα το control unit της ALU. Ως είσοδοι στη ALU πλέον ορίζουμε τον καταχωρητή Rf[A]=Rf[rd] και το Immediate με το κατάλληλο extension στα msb bits του.Η επιλογή του Immediate ελένχεται απο το control ενός πολυπλέκτη(Συγκεκριμένα ALU_bin_sel=1).Το αποτέλεσμα της ALU ακολουθεί την ίδια σκέψη με τις *R-type εντολές*.
- Branch-εντολές: Στις εντολές αυτές υπάρχει μια αισθητή διαφοροποίηση στο IFSTAGE τμήμα του datapath. Συγκεκριμένα εφόσον λάβουμε από την μνήμη μια τέτοια εντολή τότε πηγαίνουμε στη Decode κατάσταση. Από εκεί λαμβάνουμε το opcode κομμάτι. Αν αυτό προσδιορίζει μια “b” εντολή τότε πηγαίνουμε απευθείας στην κατάσταση b όπου και εγγράφουμε πλεόν στον καταχωρητή PC(PC_ldEn=1) το PC=PC+4 αλλά το PC=PC+4+PCimmed. Η διαφοροποίηση αυτή γίνεται με πολυπλέκτη όπου το control γίνεται PC_sel=1. Στην συνέχεια η fsm δεν περνάει από την κατάσταση InstructionFetch αλλά πηγαίνει κατευθείαν στην InstructionDelay για την απαραίτητη καθυστέριση της μνήμης εντολών για δέσμευση της επόμενης εντολής. Οι εντολές “bne” και “beq” διαφοροποιούνται με την “b” στο γεγονός ότι θα πρέπει να γίνει σύγκριση καταχωρητών για το αν θα μεταβούμε στην εντολή που επιλέγεται. Για την σύγκριση, δεσμεύουμε τους ζητούμενους καταχωρητές από την Register-file (Συγκεκριμένα τους καταχωρητές Rf[A] και Rf[B] που έχουν ενημερωθεί στην Decode κατάσταση) και εκτελούμε μια εντολή subtraction στην ALU. Για να εκτελεστεί η πράξη στην ALU με τους ενδιάμεσους καταχωρητές χρειάζεται ένας ακόμη κύκλος ρολογιού και γι αυτό προσθέσαμε μια ακόμη κατάσταση(beqDelay και bneDelay αντίστοιχα). Το αποτέλεσμα της ALU θα

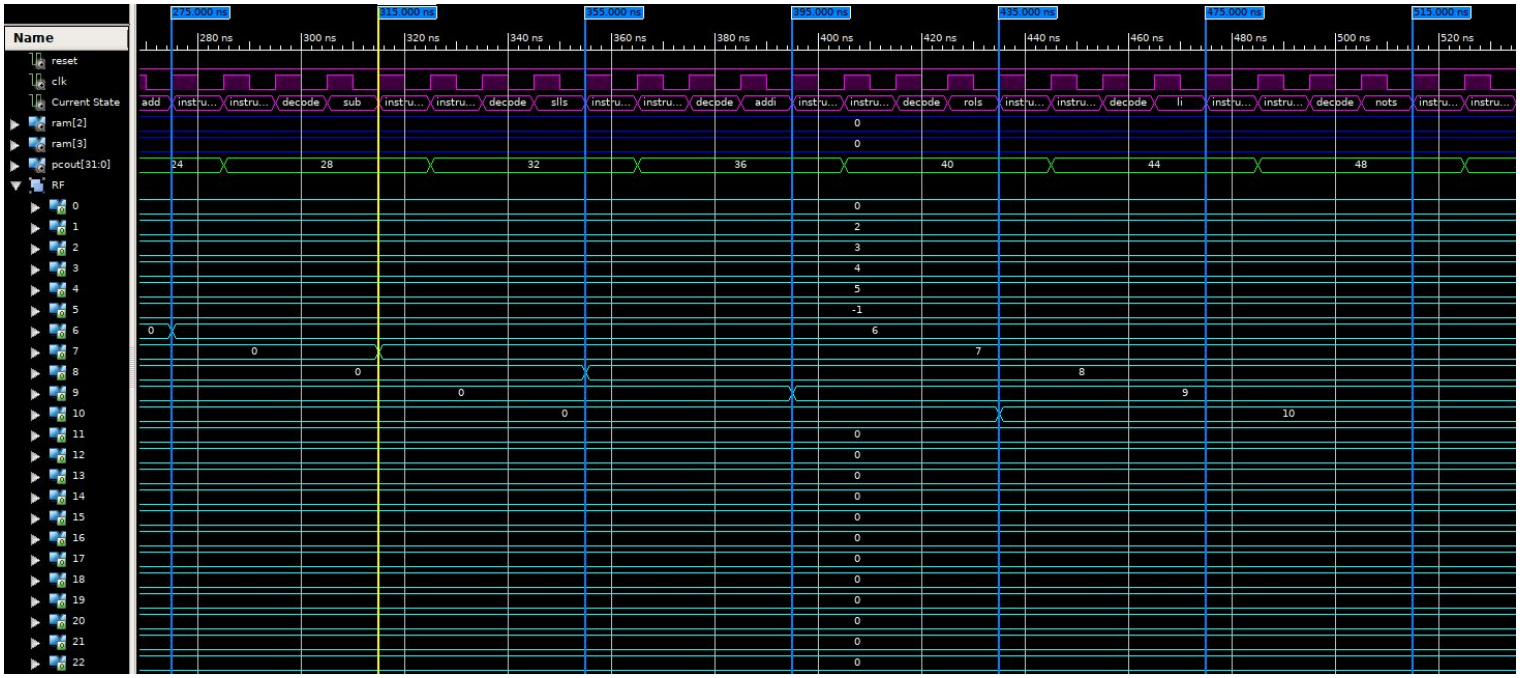
ενεργοποιεί την έξοδο της ALU για μηδενικό αποτέλεσμα που σημαίνει ότι όντως οι καταχωρητές είναι ίσοι η δεν θα το ενεργοποιεί και δεν θα είναι ίσοι. Ανάλογα το αποτέλεσμα η επόμενη κατάσταση θα είναι για ισότητα η b(όπου περιγράψαμε παραπάνω την ροή της) ή η κατάσταση InstructionFetch.

- *Store-load εντολές:* Οι εντολές αυτές διαφοροποιούνται ελάχιστα με τις I-type εντολές. Συγκεκριμένα η δέσμευση εντολής από την μνήμη είναι η ίδια όπως και το decode κομμάτι για την ενημέρωση των καταχωρητή RF[rs] και του Immediate. Στην συνέχεια κάθε τέτοια εντολή θέλει την πράξη της πρόσθεσης συνεπώς χρειάζεται την ALU. Ως τελεστές της πρόσθεσης χρειάζονται ο καταχωρητής RF[rs] και το όποιο extension του Immediate (ALU_bin_sel=1). Φυσικά ο καταχωρητής RF[rs] έχει προέλθει από τον ενδιάμεσο καταχωρητή RF[A] (άρα RF_AB_sel=1). Το group αυτών των εντολών αναφέρεται στις sw, sb, lw, lb. Θα πρέπει να αποφασισθεί αν θα γίνει δηλαδή store ή load ενός byte ή μιας ολόκληρης λέξης (word). Αυτό επιτυγχάνεται με την χρήση δύο όμοιων πολυπλεκτών 2*32bit εισόδων και 1*32 bit εξόδου μέσα στο MEMSTAGE.

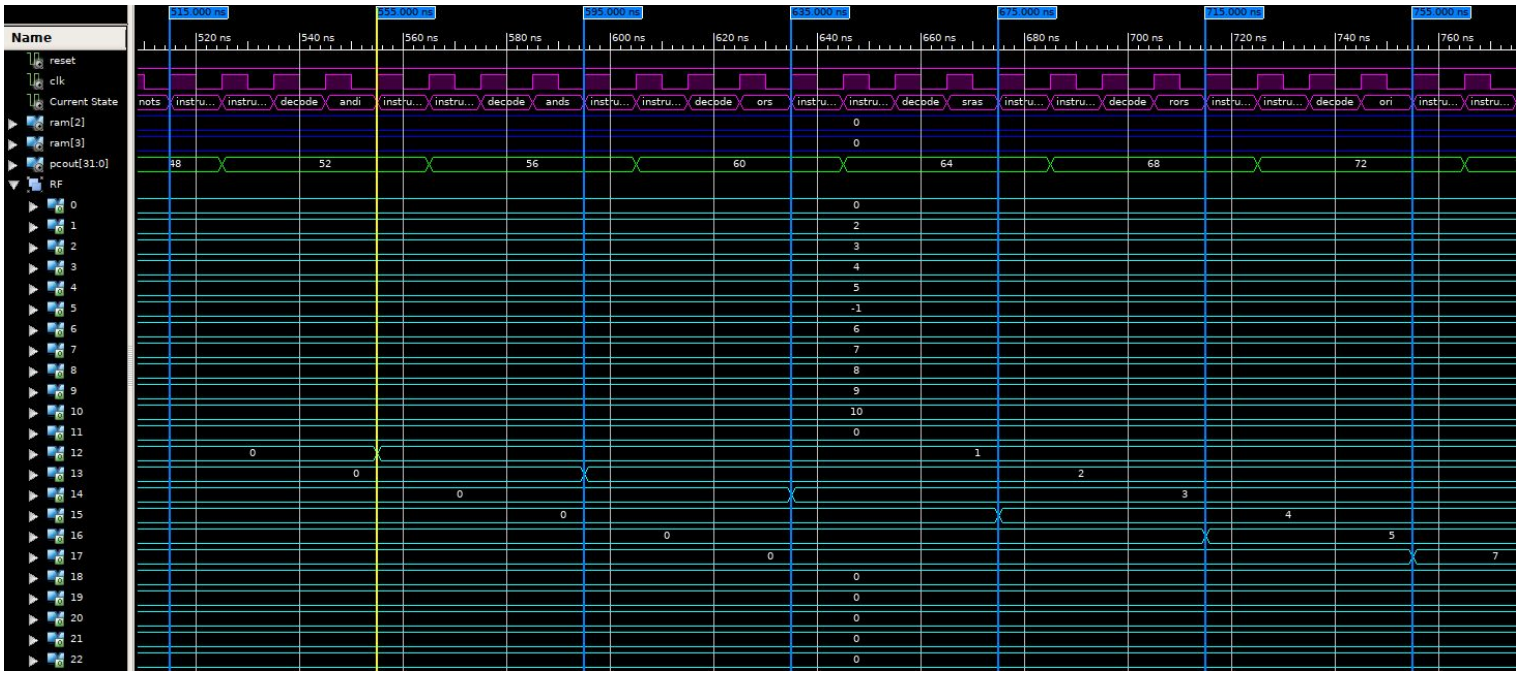
Κυμματομορφές



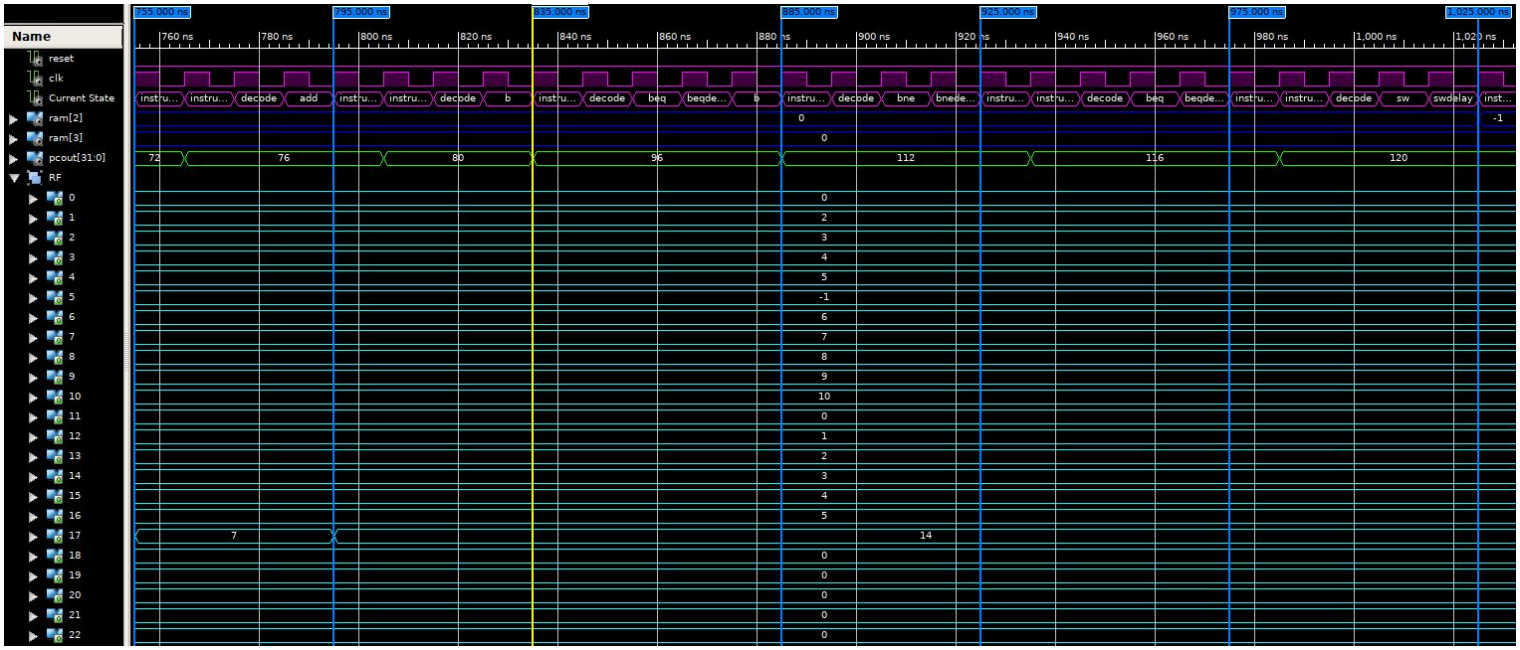
- 0-35: nop
- 35-75: li $1, 2 -- $1 = 2
- 75-115: li $2, 3 -- $2 = 3
- 115-155: li $3, 4 -- $3 = 4
- 155-195: li $4, 5 -- $4 = 5
- 195-235: li $5, -1 -- $5 = -1
- 235-275: add $6, $2, $2 -- $6 = 6



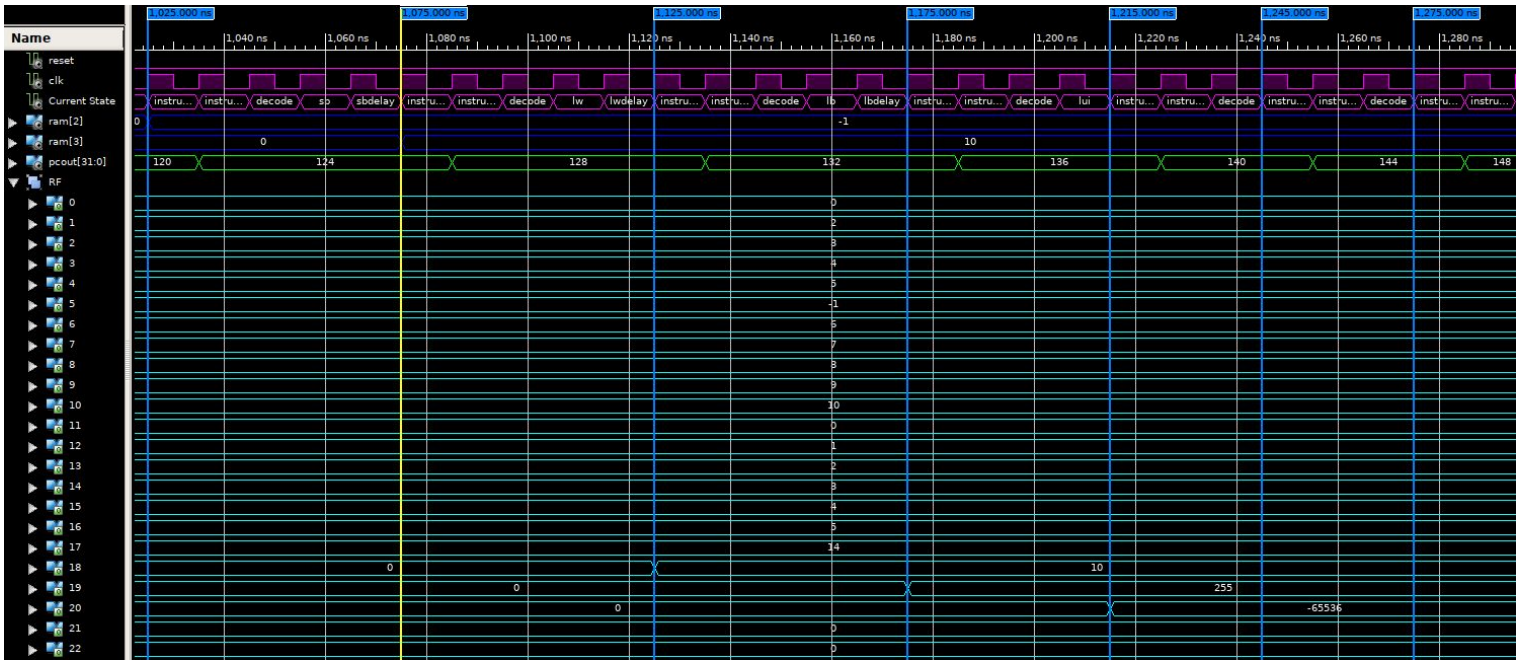
- 275-315: sub $7, $6, $5 -- $7 = 7
- 315-355: shl $8, $3 -- $8 = 8
- 355-395: addi $9, $4, 4 -- $9 = 9
- 395-435: rol $10, $4 -- $10 = 10
- 435-475: li $0, 11 -- NOP!!!!!!!
- 475-515: not $11, $5 -- $11 = 0



- 515-555: andi $12, $7, 1 -- $12 = 1
- 555-595: and $13, $1, $5 -- $13 = 2
- 595-635: or $14, $12, $1 -- $14 = 3
- 635-675: shr $15, $8 -- $15 = 4
- 675-715: ror $16, $10 -- $16 = 5
- 715-755: ori $17, $4, 2 -- $17 = 7, PC = 72



- 755-795: add $17, $17, $17 -- $17 = 14
- 795-835: b 3 -- PC = PC + 4 + 12 = 96
- 835-885: beq $14, $2, 3 -- PC = PC + 4 + 12 = 112
- 885-925: bne $14, $2, 16 -- PC = 116
- 925-975: beq $15, $10, 16 -- PC = 120
- 975-1025: sw $5, 4($3) -- MEM[2] = -1



- 1025-1075: sb $10, 8($15) -- MEM[3] = 10
- 1075-1125: lw $18, 2($10) -- $18 = MEM[3] = 10
- 1125-1175: lb $19, 8($0) -- $19 = MEM[2] = 255
- 1175-1215: lui $20, -1 -- $20 = -65536

Συμπεράσματα-Παρατηρήσεις

Μία σημαντική παρατήρηση είναι ότι στην εντολή lui(load upper immediate) η οποία επεξεργάζεται μια σταθερά και την γράφει σε έναν καταχωρητή της Register-file δεν υπάρχει δυνατότητα στον υπάρχων πολυπλέκτη που βρίσκεται στην WrDataIn του Register File να λάβει την “επεξεργασμένη” σταθερά καθώς δέχεται εισόδους μόνο από αποτέλεσμα ALU (alu_out) και από μνήμη(mem_out). Συνεπώς στον υπάρχων πολυπλέκτη προσθέσαμε ένα ακόμα bit στο control unit του για να μπορεί να δέχεται 3 εισόδους με την τρίτη να είναι το αποτέλεσμα του immediate extender.