

Έκδοση RISC_V 2025

Κεφάλαιο 7

Μικροαρχιτεκτονική

Καθηγητής Αντώνης Πασχάλης



dscal

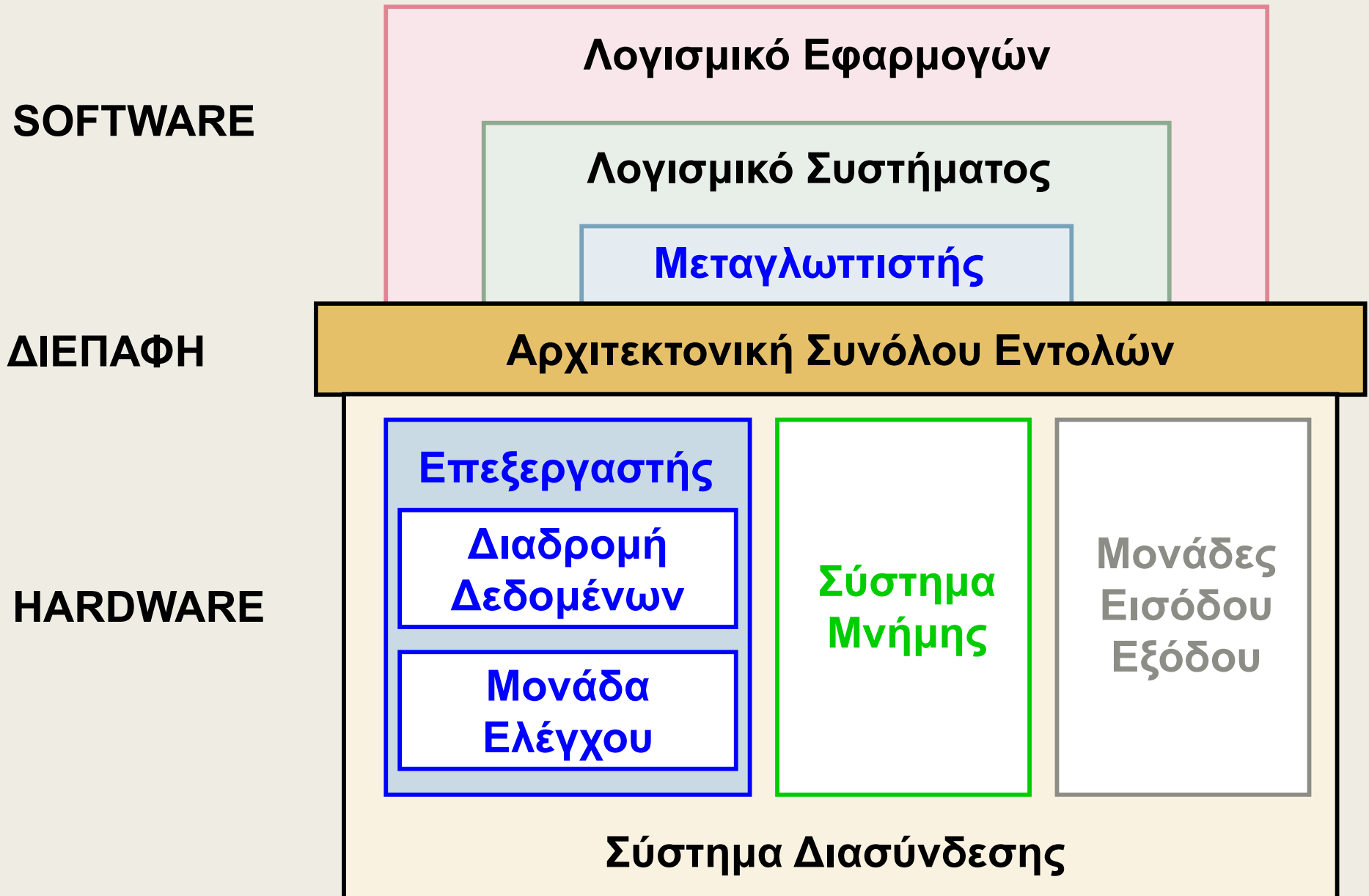
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Περιεχόμενα κεφαλαίου 7

- Εισαγωγή
- Αρχιτεκτονικοί καταχωρητές
- Βασικές μικροαρχιτεκτονικές RISC-V
- Σχεδίαση μικροαρχιτεκτονικής ενός κύκλου, πολλών κύκλων και με διοχέτευση
 - διαδρομή δεδομένων (*datapath*)
 - μονάδα ελέγχου (*control unit*)
 - ανάλυση επιδόσεων
 - συγκρίσεις

Λογισμικό εφαρμογών	
Λειτουργικά συστήματα	
Αρχιτεκτονική	
Μικρο-αρχιτεκτονική	
Λογική	
Ψηφιακά κυκλώματα	
Αναλογικά κυκλώματα	
Διατάξεις	
Φυσική	

Ο υπολογιστής



Μικροαρχιτεκτονική (microarchitecture)

- Προσδιορίζει **πως υλοποιείται η αρχιτεκτονική στο υλικό**
 - Σχεδίαση της **διαδρομής δεδομένων** (datapath) του επεξεργαστή
 - χωροταξική διάταξη των καταχωρητών, των μνημών, των μονάδων ALU και των άλλων δομικών στοιχείων που απαρτίζουν τη διαδρομή δεδομένων
 - Σχεδίαση της **μονάδας ελέγχου** (control unit) του επεξεργαστή
 - παραγωγή των κατάλληλων σημάτων ελέγχου για τον χρονισμό του επεξεργαστή και των υπολοίπων διατάξεων με τη χρήση:
 - **συνδυαστικής λογικής** (στην περίπτωση επεξεργαστών ενός κύκλου και επεξεργαστών με διοχέτευση) ή
 - **μηχανών πεπερασμένων καταστάσεων** (στην περίπτωση επεξεργαστών πολλών κύκλων)
- Συχνά υπάρχουν πολλές διαφορετικές μικροαρχιτεκτονικές για την ίδια αρχιτεκτονική με διαφορετικούς συμβιβασμούς μεταξύ των τριών βασικών αξόνων σχεδίασης:
 - **επιδόσεις** (χρόνος εκτέλεσης προγράμματος)
 - **κόστος** (απαιτούμενοι πόροι, κατανάλωση ισχύος)
 - **πολυπλοκότητα** στη σχεδίαση

Αρχιτεκτονική κατάσταση

- Η **αρχιτεκτονική υπολογιστών** ορίζεται από το **σύνολο εντολών** της και από την **αρχιτεκτονική κατάσταση** της
- Η **αρχιτεκτονική κατάσταση** του επεξεργαστή RISC-V συνίσταται σε:
 - Σε ένα σύνολο 32 καταχωρητών που αποθηκεύονται σε **32 καταχωρητές μεγέθους 32 bit** (το αρχείο καταχωρητών)
 - έναν **μετρητή PC (program counter)**
- Οι καταχωρητές που συνιστούν την αρχιτεκτονική κατάσταση ονομάζονται **αρχιτεκτονικοί καταχωρητές** (architectural registers)
 - Κάθε μικροαρχιτεκτονική RISC-V πρέπει να υλοποιεί **όλους** τους αρχιτεκτονικούς καταχωρητές
- Κάποιες μικροαρχιτεκτονικές περιλαμβάνουν πρόσθετους καταχωρητές, τους **μη αρχιτεκτονικούς καταχωρητές** (nonarchitectural registers)
 - για να ελαχιστοποιήσουν τη λογική ή για να βελτιώσουν τις επιδόσεις

τρέχουσα αρχιτεκτονική
κατάσταση

εκτέλεση εντολής
→
σύνολο δεδομένων

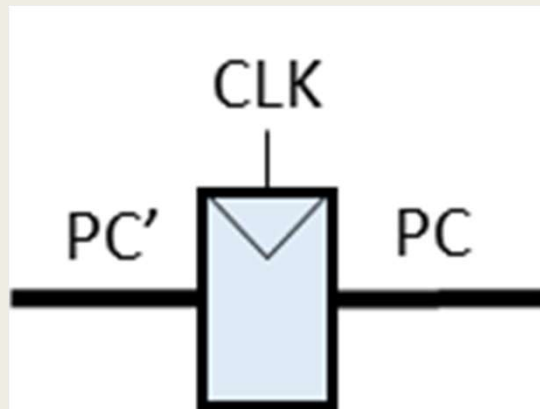
επόμενη αρχιτεκτονική
κατάσταση

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Αρχικά, ορίζουμε μία μικροαρχιτεκτονική που απαρτίζεται από δύο διακριτές μονάδες που αλληλοεπιδρούν μεταξύ τους:
 - τη **διαδρομή δεδομένων** (*datapath*) των 32 bit, που εκτελεί λειτουργίες με λέξεις δεδομένων και περιέχει υπομονάδες όπως:
 - διακριτές μνήμες εντολών και δεδομένων
 - αρχιτεκτονικούς και μη καταχωρητές
 - μονάδα ALU για την εκτέλεση πράξεων
 - πολυπλέκτες για τον έλεγχο της ροής δεδομένων
 - επιπλέον συνδυαστική λογική για τον υπολογισμό της επόμενης αρχιτεκτονικής κατάστασης και την επέκταση πρόσημου/μηδενός
 - τη **μονάδα ελέγχου** (*control unit*) που δίνει οδηγίες στη διαδρομή δεδομένων σχετικά με το πώς να εκτελέσει κάθε εντολή
 - παράγοντας και συγχρονίζοντας (όπου απαιτείται) τα κατάλληλα σήματα ελέγχου των υπομονάδων της διαδρομής δεδομένων
 - λαμβάνοντας υπόψη τα πεδία δήλωσης κωδικού λειτουργίας (op) και επεκτάσεων του (funct3 και funct7) της εντολής

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

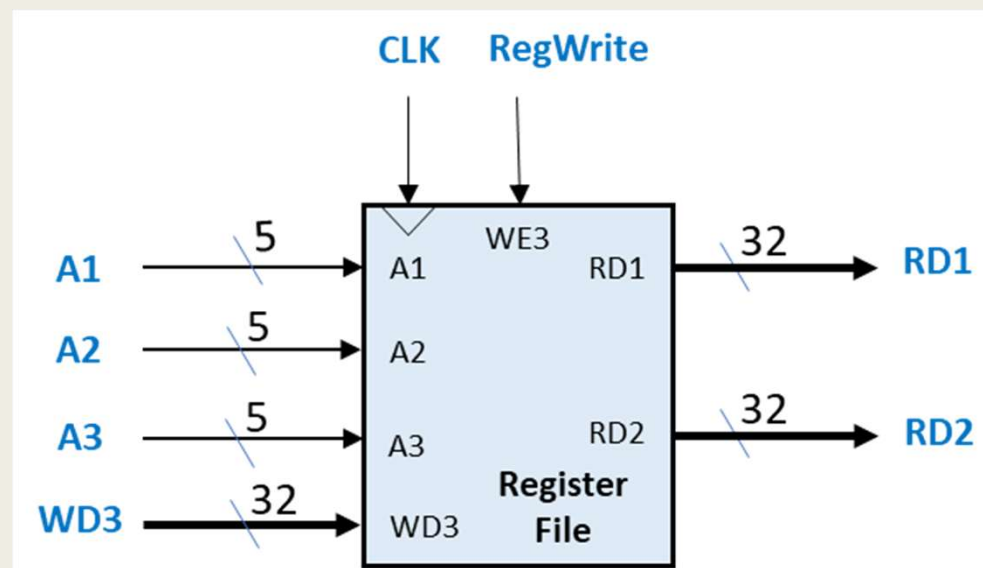
- Στη συνέχεια, ξεκινάμε τη σχεδίαση της διαδρομής δεδομένων των 32 bit στο **επίπεδο μεταφοράς καταχωρητή (RTL)** με τον ορισμό των **αρχιτεκτονικών καταχωρητών** του επεξεργαστή
- **Μετρητής προγράμματος** (program counter - PC) των 32 bit
 - η έξοδός του (PC) δείχνει τη διεύθυνση της τρέχουσας εντολής
 - η είσοδός του (PC') υποδεικνύει τη διεύθυνση της επόμενης εντολής
 - αρχικοποιείται στην τιμή 0 με το σήμα RESET (δεν φαίνεται στο σχήμα)



Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

■ Αρχείο καταχωρητών (register file - RF)

- περιέχει 32 καταχωρητές των 32 bit (x0-x31)
- διαθέτει δύο θύρες ανάγνωσης (A1/RD1 και A2/RD2) και μία θύρα εγγραφής (A3/WD3)
- καθεμία από τις διευθύνσεις των 5 bit (A1, A2 και A3) μπορεί να προσπελάσει και τους 32 καταχωρητές
- οι θύρες 1 και 2 διαβάζουν ασύγχρονα τα δεδομένα δύο καταχωρητών A1 και A2 και τα μεταφέρουν στις εξόδους RD1 και RD2, αντίστοιχα
- η θύρα 3 εγγράφει σύγχρονα (κατά την ανερχόμενη ακμή του ρολογιού) τα δεδομένα από την είσοδο WD3 στον καταχωρητή A3, όταν το σήμα ελέγχου WE3 έχει την τιμή 1



Σύνολο καταχωρητών RISC-V

- Αποθηκεύονται στο αρχείο 32 καταχωρητών των 32 bit

Όνομασία	Αρ. καταχωρητή	Χρήση
zero	x0	Σταθερή τιμή 0 (Δεν εγγράφεται)
ra	x1	Επιστρεφόμενη τιμή (return address)
sp	x2	Δείκτης (κορυφής) στοίβας (stack pointer)
gp	x3	Καθολικός δείκτης (global pointer)
tp	x4	Δείκτης νήματος (thread pointer)
t0-t2	x5-7	Προσωρινοί (temporary) καταχωρητές t0-t2
s0/fp	x8	Αποθηκευμένος (saved) καταχωρητής s0 ή Δείκτης πλαισίου (στοίβας) (frame pointer)
s1	x9	Αποθηκευμένος καταχωρητής s1
a0-a1	x10-11	Ορίσματα συναρτήσεων/Επιστρεφόμενες τιμές
a2-a7	x12-17	Ορίσματα συναρτήσεων
s2-s11	x18-27	Αποθηκευμένοι καταχωρητές s2-s11
t3-t6	x28-31	Προσωρινοί καταχωρητές t3-t6

Στους αποθηκευμένους καταχωρητές (s) κρατάμε τις μεταβλητές, ενώ στους προσωρινούς καταχωρητές (t) κρατάμε τα ενδιάμεσα αποτελέσματα των πράξεων

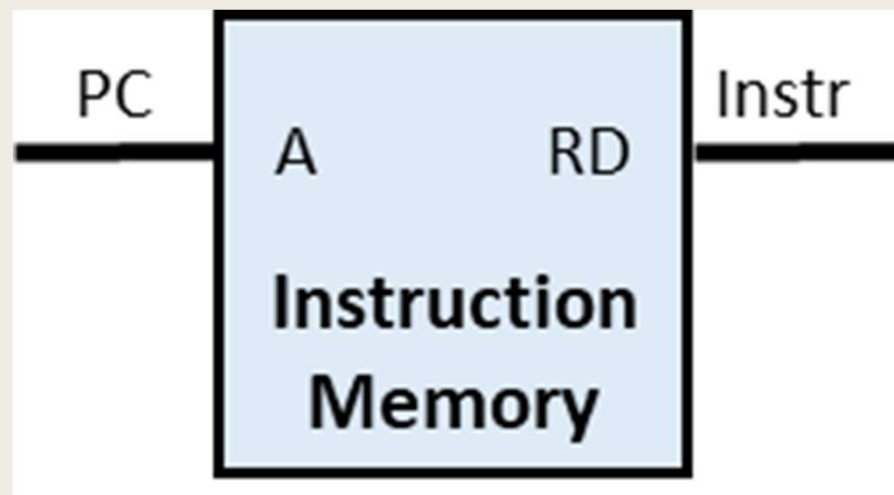
Προγραμματισμός – Κλήσεις συναρτήσεων

- Η αρχιτεκτονική RISC-V χωρίζει τους καταχωρητές του αρχείου καταχωρητών σε δύο κατηγορίες, ώστε να αποφεύγονται άσκοπες αποθηκεύσεις στη στοίβα
 - οι **διατηρούμενοι καταχωρητές** αποθηκεύονται από την **καλούμενη συνάρτηση**, όταν τους χρησιμοποιεί
 - οι **μη διατηρούμενοι καταχωρητές** αποθηκεύονται από την **καλούσα συνάρτηση**, όταν τους χρησιμοποιεί

Διατηρούμενοι	Μη Διατηρούμενοι
Αποθηκευόμενοι καταχωρητές: s0–s11	Προσωρινοί καταχωρητές: t0–t6
Δείκτης στοίβας: sp	Καταχωρητές ορισμάτων: a0–a7
Διεύθυνση επιστροφής: ra	
Στοίβα πάνω από τον δείκτη στοίβας	Στοίβα κάτω από τον δείκτη στοίβας

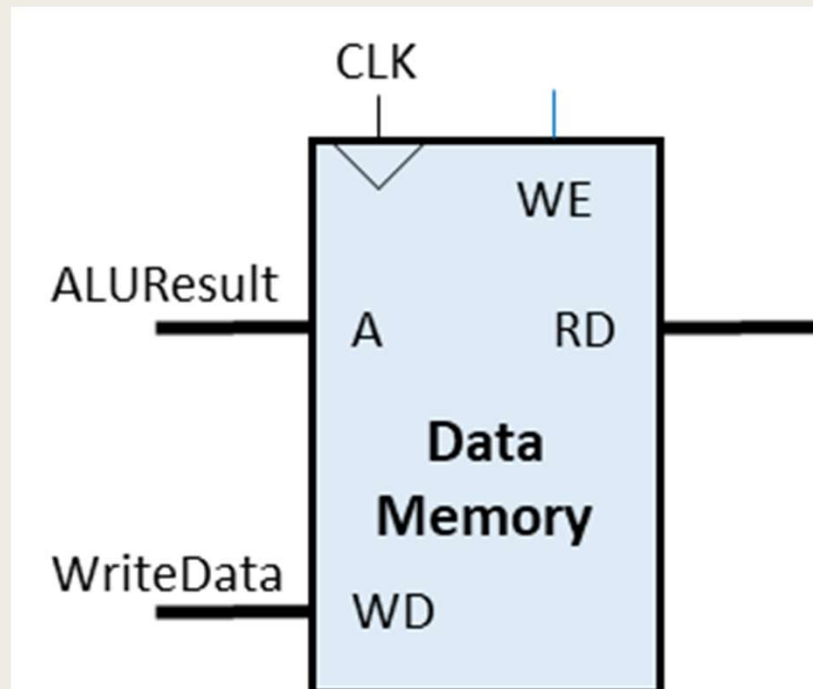
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, προχωράμε τη σχεδίαση της διαδρομής δεδομένων των 32 bit στο **επίπεδο μεταφοράς καταχωρητή (RTL)** με τον ορισμό των **μνημών εντολών και δεδομένων** του επεξεργαστή
- **Μνήμη εντολών** (instruction memory - IM) των 2^N λέξεων των 32 bit
 - δέχεται ως είσοδο (*A*) τη διεύθυνση των *N bit* της τρέχουσας εντολής
 - η διεύθυνση της τρέχουσας εντολής είναι αποθηκευμένη στον μετρητή προγράμματος, ισχύει: $A(N-1:0) = PC(N+1:2)$
 - μεταφέρει στην έξοδο δεδομένων ανάγνωσης (*RD*) την εντολή (*Instr*) σε γλώσσα μηχανής των 32 bit που είναι αποθηκευμένη στη διεύθυνση *A*
 - στην απλουστευμένη σχεδίαση αντιμετωπίζεται ως **μνήμη ROM** (συνδυαστικό κύκλωμα)



Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- **Μνήμη δεδομένων** (data memory - DM) των 2^M λέξεων των 32 bit
 - δέχεται ως είσοδο (A) τη διεύθυνση των M bit των δεδομένων που διαβάζονται ή γράφονται στη μνήμη δεδομένων
 - η διεύθυνση δεδομένων υπολογίζεται στη μονάδα ALU, ισχύει $A(M-1:0) = \text{ALUResult}(M+1:2)$
 - διαθέτει μία θύρα είτε για ανάγνωση (A/RD) είτε για εγγραφή (A/WD)
 - η ανάγνωση γίνεται σύγχρονα (Block RAM) ή ασύγχρονα (Distributed RAM)
 - η εγγραφή γίνεται σύγχρονα, όταν το σήμα ελέγχου WE έχει την τιμή 1



Στην περίπτωση της μικροαρχιτεκτονικής του ενός κύκλου η ανάγνωση γίνεται μόνο ασύγχρονα με Distributed RAM

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποια από τις βασικές μικροαρχιτεκτονικές θα υλοποιήσουμε:
 - έχουμε τρεις επιλογές διαφορετικής πολυπλοκότητας από την πιο απλή μέχρι την πιο σύνθετη
 - για εκπαιδευτικούς λόγους ξεκινάμε από την πιο απλή
- **Μικροαρχιτεκτονική ενός κύκλου**
 - διαδρομή δεδομένων ενός κύκλου
 - μονάδα ελέγχου ως συνδυαστική λογική
 - χρήση μόνο αρχιτεκτονικών καταχωρητών
 - απαίτηση ξεχωριστών μνημών για εντολές και δεδομένα
 - ολόκληρη η εντολή εκτελείται σε έναν κύκλο ρολογιού
 - η περίοδος του CLK προσδιορίζεται από την εντολή LW (load word) που έχει τη μεγαλύτερη καθυστέρηση διάδοσης

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

■ Μικροαρχιτεκτονική πολλών κύκλων

- διαδρομή δεδομένων πολλών κύκλων
- μονάδα ελέγχου ως μηχανή πεπερασμένων καταστάσεων (FSM)
- χρήση αρχιτεκτονικών, αλλά και μη αρχιτεκτονικών καταχωρητών
 - προσωρινοί καταχωρητές που τοποθετούνται ανάμεσα σε τμήματα συνδυαστικής λογικής, ώστε να διαιρεθεί η συνδυαστική λογική της διαδρομής δεδομένων σε μικρότερα στάδια τα οποία μπορούν να εκτελούνται με γρηγορότερο CLK
- δυνατότητα κοινής μνήμης για εντολές και δεδομένα
 - οι ξεχωριστές μνήμες προσφέρουν λιγότερες αλλαγές στη διαδρομή δεδομένων
- κάθε φορά εκτελείται μόνο μία εντολή, αλλά κάθε εντολή χρειάζεται περισσότερους από έναν κύκλους του ρολογιού για να ολοκληρωθεί
 - οι απλούστερες εντολές εκτελούνται σε λιγότερους κύκλους από ό,τι οι πιο περίπλοκες (π.χ. η εντολή LW σε 5 κύκλους)
 - η περίοδος του CLK προσδιορίζεται από **το πιο αργό στάδιο**
- χρησιμοποιείται σε απλούς ενσωματωμένους επεξεργαστές

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

■ Μικροαρχιτεκτονική με διοχέτευση (pipeline)

- διαδρομή δεδομένων με διοχέτευση
- μονάδα ελέγχου ως συνδυαστική λογική, αλλά με χρονισμό των σημάτων ελέγχου μέσω καταχωρητών διοχέτευσης
- χρήση αρχιτεκτονικών, αλλά και μη αρχιτεκτονικών καταχωρητών
 - καταχωρητές διοχέτευσης που τοποθετούνται ανάμεσα σε τμήματα συνδυαστικής λογικής, ώστε να διαιρεθεί η συνδυαστική λογική της διαδρομής δεδομένων σε μικρότερα στάδια τα οποία μπορούν να εκτελούνται με γρηγορότερο CLK
- απαίτηση ξεχωριστών μνημών για εντολές και δεδομένα
- σε κάθε στάδιο της διαδρομής δεδομένων εκτελείται **άλλη εντολή**
 - πολλές εντολές εκτελούνται ταυτόχρονα, άρα βελτιώνεται σημαντικά η διεκπεραιωτική ικανότητα (throughput)
 - όλες οι εντολές εκτελούνται σε τόσους κύκλους όσο είναι και τα διαφορετικά στάδια της διαδρομής δεδομένων (συνήθως σε 5 κύκλους)
 - η περίοδος του CLK προσδιορίζεται από **το πιο αργό στάδιο**
- απαίτηση για προσθήκη λογικής για το **χειρισμό των εξαρτήσεων δεδομένων και ελέγχου** μεταξύ ταυτόχρονα εκτελούμενων εντολών
- χρησιμοποιείται στους περισσότερους επεξεργαστές σήμερα

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου R (register)

31:25	24:20	19:15	14:12	11:7	6:0
funct7	rs2	rs1	funct3	rd	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

- **ADD** $Rd, Rs1, Rs2$ $Rd = Rs1 + Rs2$
- **SUB** $Rd, Rs1, Rs2$ $Rd = Rs1 - Rs2$
- **AND** $Rd, Rs1, Rs2$ $Rd = Rs1 \text{ and } Rs2$
- **OR** $Rd, Rs1, Rs2$ $Rd = Rs1 \text{ or } Rs2$
- **XOR** $Rd, Rs1, Rs2$ $Rd = Rs1 \text{ xor } Rs2$
- **SLT** $Rd, Rs1, Rs2$ $Rd = 1 \text{ if } rs1 < rs2, \text{ else } 0$
- **SLTU** $Rd, Rs1, Rs2$ $Rd = 1 \text{ if } rs1 < rs2, \text{ else } 0$
- **SLL** $Rd, Rs1, Rs2$ $Rd = Rs1 \ll Rs2(4:0)$
- **SRL** $Rd, Rs1, Rs2$ $Rd = Rs1 \gg Rs2(4:0)$
- **SRA** $Rd, Rs1, Rs2$ $Rd = Rs1 \ggg Rs2(4:0)$

Αρχικά δεν θα υλοποιήσουμε τις εντολές ολίσθησης

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου *I* (*immediate*)

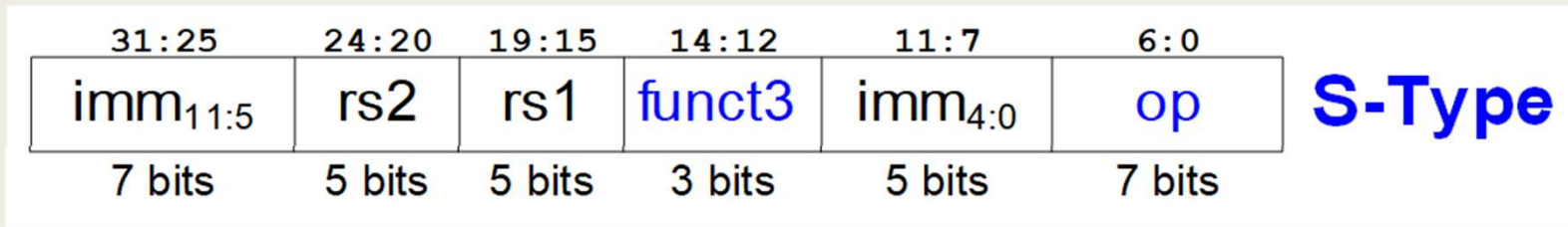
31:20	19:15	14:12	11:7	6:0
imm _{11:0}	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

- | | | | |
|---|------|---------------|---|
| ■ | ADDI | Rd, Rs1, imm | $Rd = Rs1 + S_Ext(imm)$ |
| ■ | ANDI | Rd, Rs1, imm | $Rd = Rs1 \text{ and } S_Ext(imm)$ |
| ■ | ORI | Rd, Rs1, imm | $Rd = Rs1 \text{ or } S_Ext(imm)$ |
| ■ | XORI | Rd, Rs1, imm | $Rd = Rs1 \text{ xor } S_Ext(imm)$ |
| ■ | LW | Rd, imm(Rs1) | $Rd = mem[Rs1 + S_Ext(imm)]$ |
| ■ | SLLI | Rd, Rs1, uimm | $Rd = Rs1 \ll imm(4:0)$ |
| ■ | SRLI | Rd, Rs1, uimm | $Rd = Rs1 \gg imm(4:0)$ |
| ■ | SRAI | Rd, Rs1, uimm | $Rd = Rs1 \ggg imm(4:0)$ |
| ■ | JALR | Rd, Rs1, imm | $PC = Rs1 + S_Ext(imm)$
$Rd = PC + 4$ |

Αρχικά δεν θα υλοποιήσουμε τις εντολές ολίσθησης και JALR (JR)

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

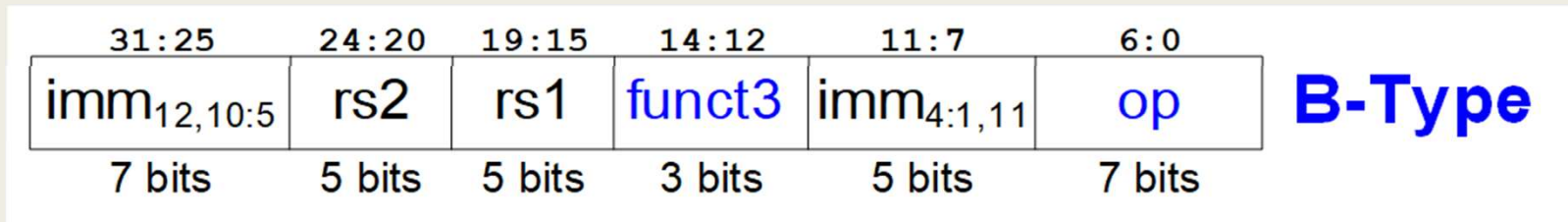
- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου *S* (store)



- **SW** **Rs2, imm(Rs1) mem[Rs1+S_Ext(imm)] = Rs2**

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου B (branch)



- BEQ Rs1, Rs2, label branch BTA if Rs1 = Rs2
- BNE Rs1, Rs2, label branch BTA if Rs1 ≠ Rs2
- BLT Rs1, Rs2, label branch BTA if Rs1 < Rs2
- BGE Rs1, Rs2, label branch BTA if Rs1 ≥ Rs2
- BLTU Rs1, Rs2, label branch BTA if Rs1 < Rs2
- BGEU Rs1, Rs2, label branch BTA if Rs1 ≥ Rs2

BTA = PC + Sign_Ext(instr(31)&instr(7)&instr(30:25)&instr(11:8)&'0')

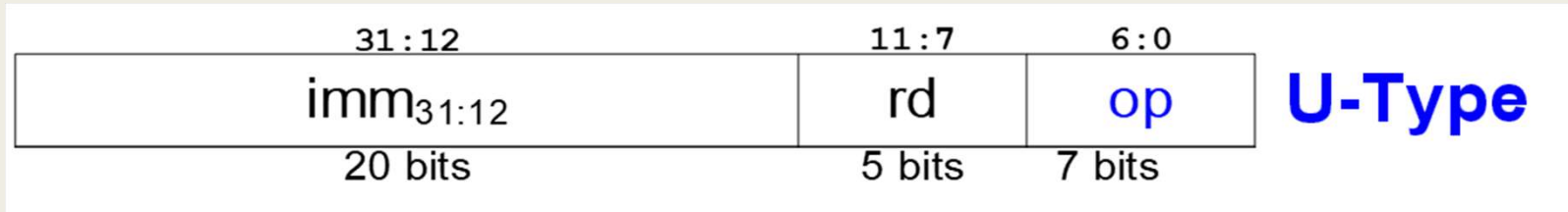
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου *B* (*branch*)
 - Εάν είναι διαθέσιμες μόνο οι εντολές SLT(U) και BEQ/BNE, τότε οι εντολές BLT(U), BGE(U), BGT(U) και BLE(U) υλοποιούνται ως εξής:

Σύγκριση	Εντολές Bxx(U)	Υλοποίηση με εντολές SLT(U) και BEQ/BNE
$s0 = s1$	BEQ s0, s1, label	BEQ s0, s1, label
$s0 \neq s1$	BNE s0, s1, label	BNE s0, s1, label
$s0 < s1$	BLT(U) s0, s1, label	SLT(U) t0, s0, s1 BNE t0, zero, label
$s0 \geq s1$	BGE(U) s0, s1, label	SLT(U) t0, s0, s1 BEQ t0, zero, label
$s0 > s1$	BGT(U) s0, s1, label	SLT(U) t0, s1, s0 BNE t0, zero, label
$s0 \leq s1$	BLE(U) s0, s1, label	SLT(U) t0, s1, s0 BNE t0, zero, label

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
 - Εντολές τύπου *U* (*upper immediate*)



■ **LUI** **Rd, upimm** **Rd = (upimm, 0x000)**

Αρχικά δεν θα υλοποιήσουμε την εντολή LUI

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Σύγκριση συνόλων βασικών εντολών αρχιτεκτονικών RISC-V και ARM:

Εντολές RISC-V	Εντολές ARM
ADD, SUB, ADDI (12 bit)	ADD(S), SUB(S) τύπου R και I (12 bit)
LW, SW	LDR, STR
AND, OR, XOR, ANDI, ORI, XORI	AND(S), ORR(S), EOR(S) τύπου R και I (12 bit)
ADDI, XORI(-1)	MOV, MVN τύπου R και I (12 bit)
NOP (ADDI, zero, 0)	NOP (MOV R0, R0)
SLT, SLTU (SLTI, SLTUI έμμεσχα)	CMP τύπου R και I (12 bit)
BEQ, BNE (BLT, BGE, BLTU, BGEU)	B, BEQ, BNE, BLT, BGE, BLO, BHS
SLL, SRL, SRA (SLLI, SRLI, SRAI)	LSL, LSR, ASR, ROR τύπου I
LUI (20 msb) (για σταθερά 32bit)	Δημιουργία σταθεράς 32 bit με περιστροφή
JALR	BL
JR	MOV PC, LR

Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- Κωδικοποίηση άμεσου τελεστέου στην αρχιτεκτονική RISC-V

funct7							4	3	2	1	0	rs1					funct3			rd					I* I S B U J
11	10	9	8	7	6	5	4	3	2	1	0	rs1					funct3			rd					
11	10	9	8	7	6	5	rs2					rs1					funct3			4	3	2	1	0	
12	10	9	8	7	6	5	rs2					rs1					funct3			4	3	2	1	11	
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	rd					
20	10	9	8	7	6	5	4	3	2	1	11	19	18	17	16	15	14	13	12	rd					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	

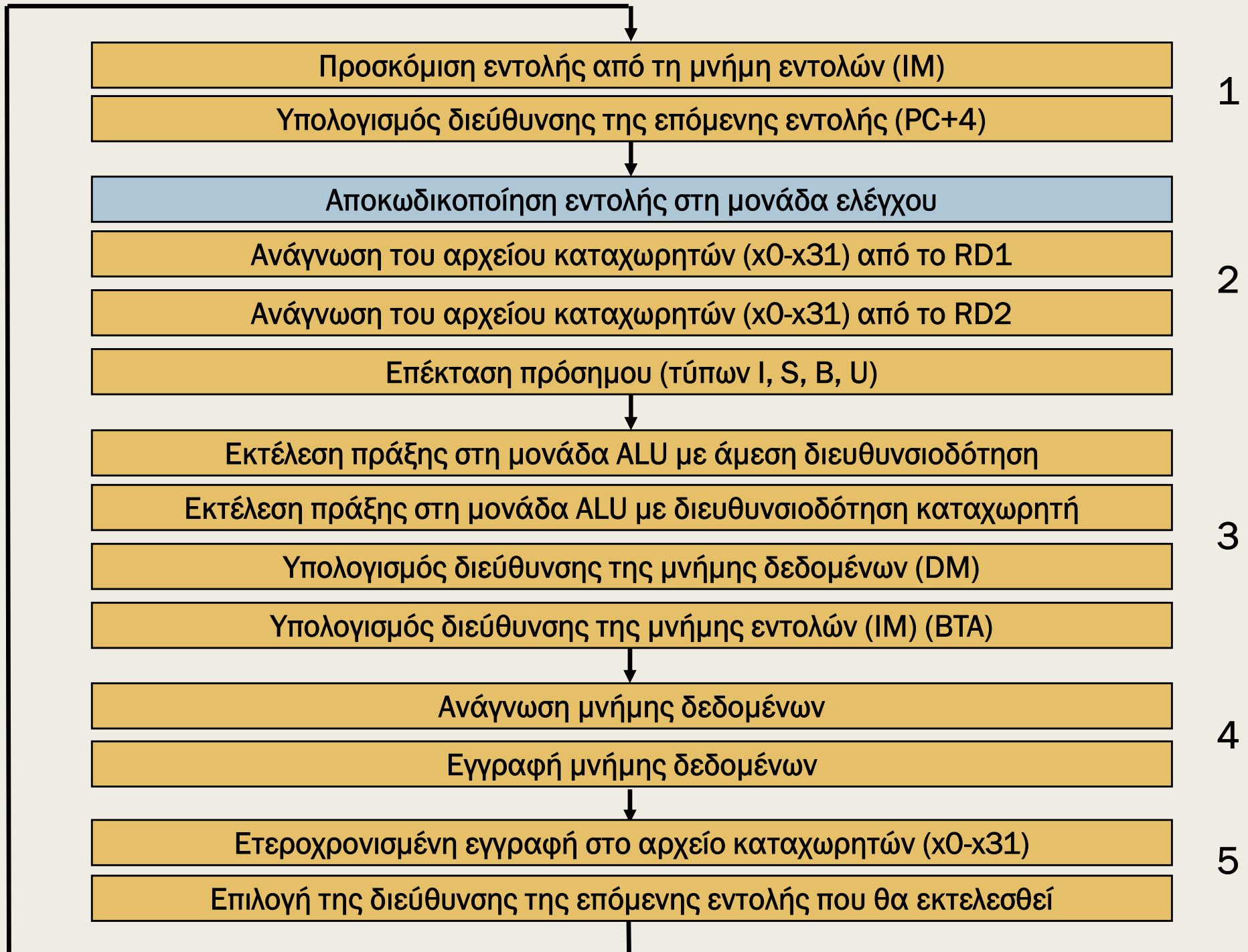
- Τα bit του άμεσου τελεστέου αποθηκεύονται στις περισσότερες των περιπτώσεων στη σειρά σε κατάλληλα πεδία της εντολής
 - Απλοποιεί το υλικό για την κατασκευή του μικροεπεξεργαστή
- Οι άμεσοι τελεστέοι υπόκεινται σε επέκταση προσήμου στα 32 bit πριν τη χρήση τους
 - το bit προσήμου του άμεσου τελεστέου βρίσκεται στο MSB της εντολής
- * Υπενθυμίζεται ότι στις ολισθήσεις τύπου I η σταθερή ποσότητα ολίσθησης καθορίζεται ως μη προσημασμένος άμεσος τελεστέος των 5 bit (uimm) στη θέση του Rs2 (24:20).
 - Ειδική περίπτωση: Δεν απαιτεί επέκταση προσήμου

Επεξεργαστής ενός κύκλου

- Στη συνέχεια, προχωρούμε στη σχεδίαση της **μικροαρχιτεκτονικής ενός κύκλου** που υλοποιεί τις εντολές που αναφέραμε
 - με βάση τους **αρχιτεκτονικούς καταχωρητές** και τις **μνήμες εντολών και δεδομένων**, προσθέτουμε την απαραίτητη **συνδυαστική λογική** μελετώντας τις **λειτουργίες** των εντολών που θα υλοποιήσουμε
 - Οι λειτουργίες ταξινομούνται σε πέντε βήματα εκτέλεσης της εντολής:
 1. Προσκόμιση εντολής και υπολογισμός της διεύθυνσης εντολής PC+4
 2. Αποκωδικοποίηση εντολής και ανάγνωση αρχείου καταχωρητών
 3. Εκτέλεση πράξεων στη μονάδα ALU
 4. Ανάγνωση ή εγγραφή στη μνήμη δεδομένων
 5. Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών και επιλογή της διεύθυνσης της επόμενης εντολής
- Κατά τη σχεδίαση προβαίνουμε σε:
 - **μελέτη διαδρομής δεδομένων**
 - **μελέτη μονάδας ελέγχου**
 - **ανάλυση επιδόσεων**

Στα σχήματα που ακολουθούν, η σχετική με τη λειτουργία ροή δεδομένων φαίνεται με **έντονο κόκκινο χρώμα**

Λειτουργίες εντολών (14)

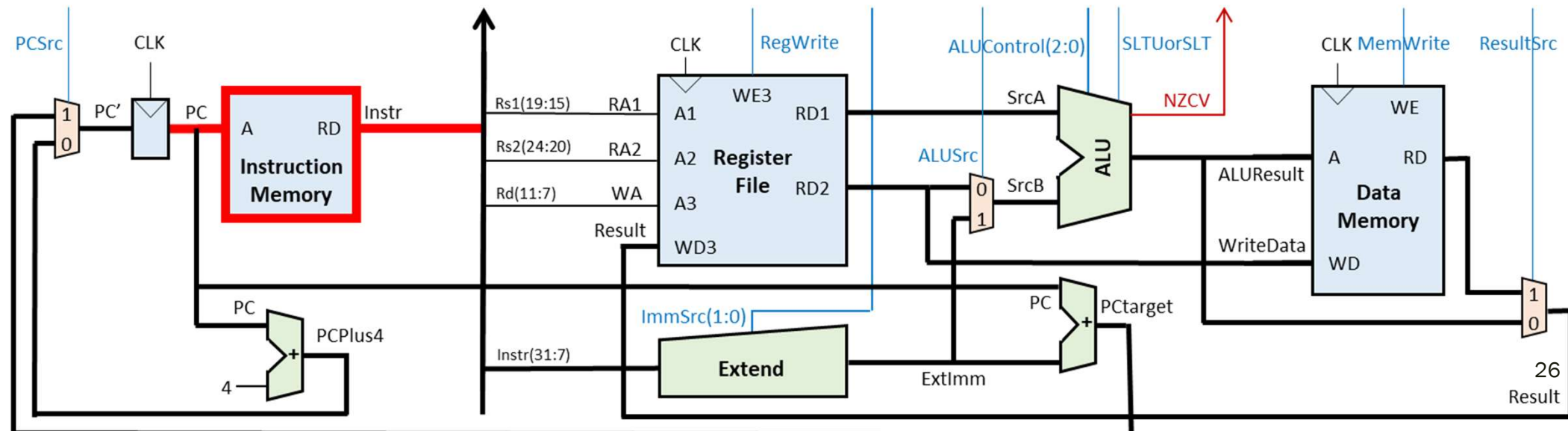


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων

Προσκόμιση εντολής από τη μνήμη εντολών (IM)

- Ο μετρητής προγράμματος PC περιέχει τη διεύθυνση της τρέχουσας εντολής και διευθυνσιοδοτεί την είσοδο διευθύνσεων A της μνήμης εντολών (IM)
- Η εντολή διαβάζεται ασύγχρονα στη θύρα ανάγνωσης RD της IM των $2^N \times 32$ bit
 - Ο μετρητής προγράμματος αποθηκεύει ευθυγραμμισμένες διευθύνσεις εντολών για μνήμες οργανωμένες σε byte, ενώ η μνήμη εντολών είναι οργανωμένη σε λέξεις
 - Ισχύει $A(N-1:0) = PC(N+1:2)$

Λειτουργία κοινή για όλες τις εντολές

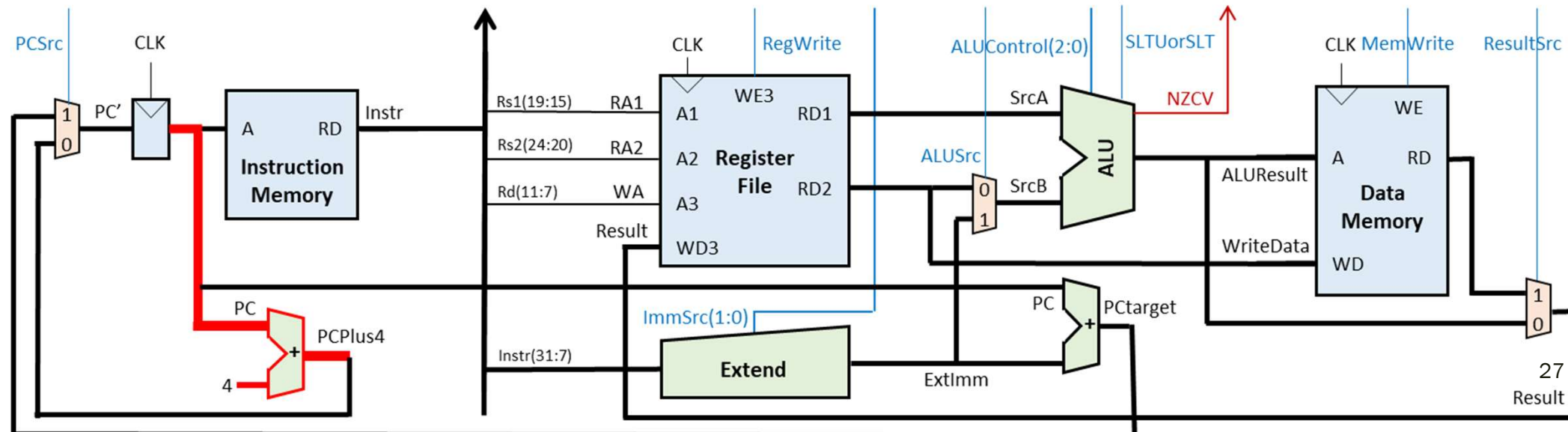


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων

Υπολογισμός διεύθυνσης της επόμενης εντολής (PC+4)

- Υπολογισμός της διεύθυνσης της αμέσως επόμενης εντολής **PC+4** (παράλληλα με την εκτέλεση της τρέχουσας εντολής)
 - σύνδεση της εξόδου PC του μετρητή προγράμματος με την είσοδο του **αθροιστή κατά 4** (incrementer)
 - εμφάνιση του αποτελέσματος της αύξησης κατά 4 στην έξοδο PCPlus4 του αθροιστή κατά 4
 - σύνδεση της εξόδου PCPlus4 με τον πολυπλέκτη επιλογής διεύθυνσης επόμενης εντολής

Λειτουργία κοινή για όλες τις εντολές



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

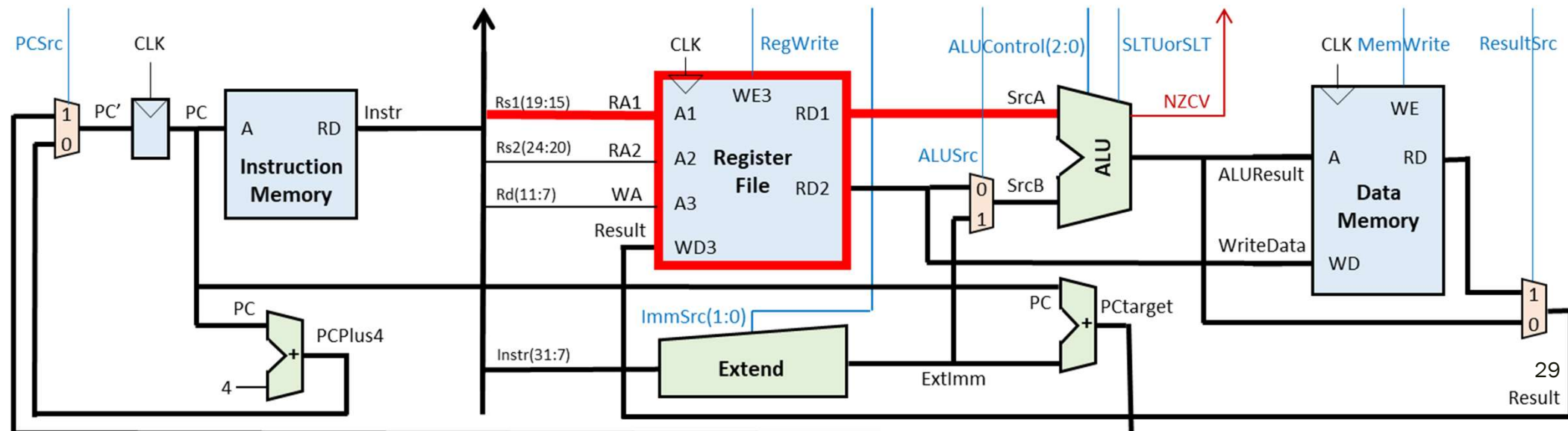
- Αρχικά, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί την εντολή **LW**
- Κώδικας συμβολικής γλώσσας:
 - **LW Rd, imm(Rs1) Rd = mem[Rs1+S_Ext(imm)]**
- Στη μορφή της εντολής ορίζονται:
 - οι διευθύνσεις του **καταχωρητή προέλευσης Rs1** και του **καταχωρητή προορισμού Rd**
 - ένας **προσημασμένος άμεσος τελεστέος imm(11:0)** των **12 bit** (*instr(31:20)*)
- Απαιτείται επέκταση προσήμου τύπου I του άμεσου τελεστέου από τα 12 bit στα 32 bit για τον υπολογισμό της σχετικής απόστασης
- Απαιτείται η εκτέλεση της πράξης της **πρόσθεσης** στη μονάδα ALU για τον προσδιορισμό της διεύθυνσης της μνήμης δεδομένων

31:20	19:15	14:12	11:7	6:0
imm _{11:0}	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

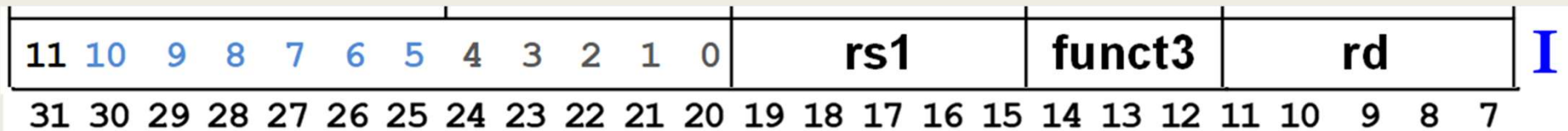
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

- Η διεύθυνση βάσης που είναι αποθηκευμένη στον **καταχωρητή προέλευσης Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)

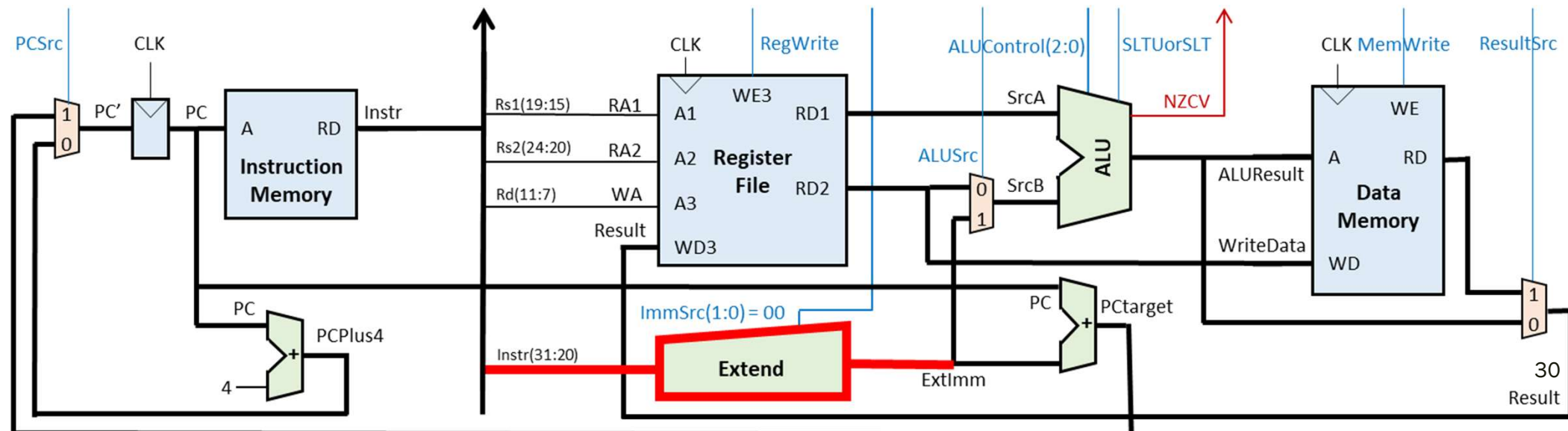


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

Επέκταση πρόσημου (τύπων I, S, B, U) – Επιλογή τύπου I



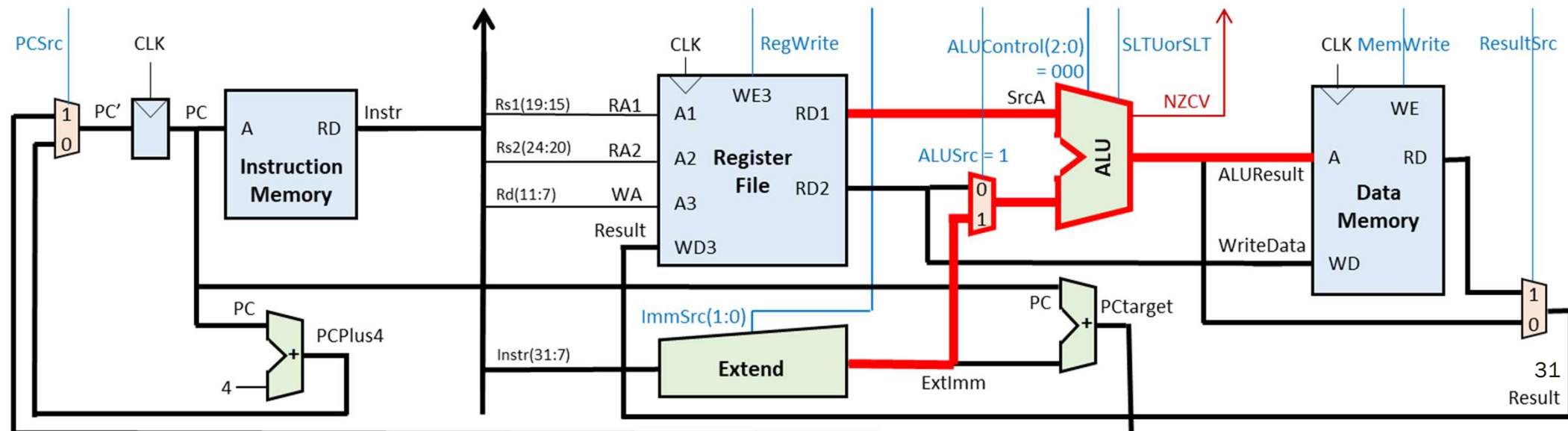
- Η εντολή LW απαιτεί και μια **σχετική απόσταση (offset)** για τον υπολογισμό της διεύθυνσης της μνήμης δεδομένων που προκύπτει με επέκταση προσήμου από τα 12 bit στα 32 bit στην μονάδα Extend
 - $\text{Extend}(31:0) = \text{Sign_Ext}(\text{instr}(31:20))$
 - Για επέκταση προσήμου τύπου I τα σήματα ελέγχου είναι: **ImmSrc(1:0) = 00**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

Υπολογισμός διεύθυνσης της μνήμης δεδομένων (DM)

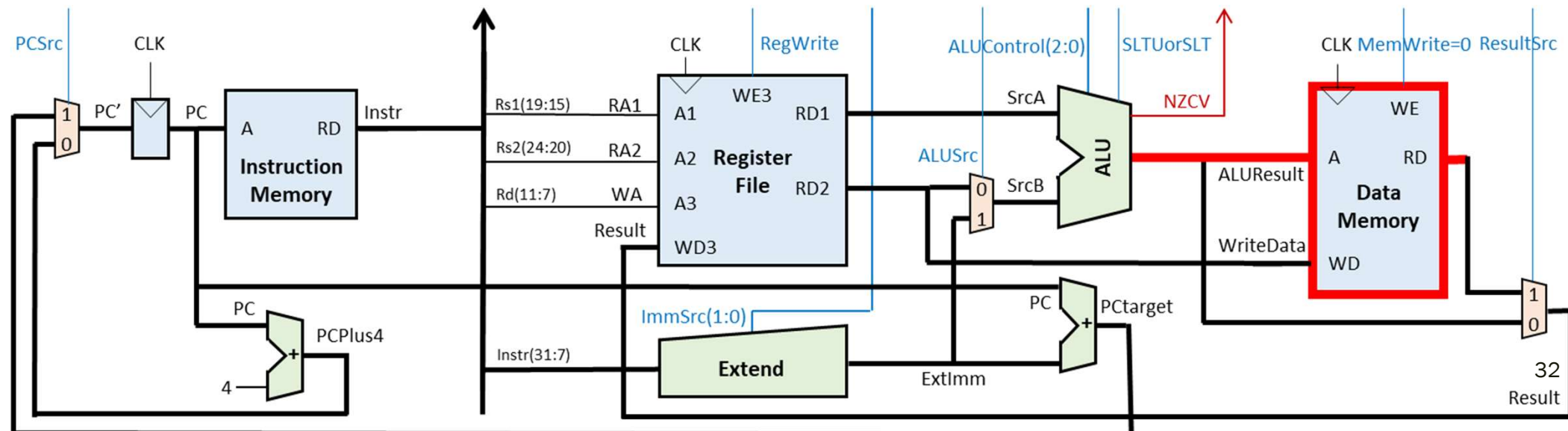
- Η **σχετική απόσταση προστίθεται** στη διεύθυνση βάσης στη μονάδα ALU, ώστε να προκύψει η διεύθυνση της μνήμης δεδομένων ως ALUResult
 - σύνδεση της θύρας *RD1* του αρχείου καταχωρητών με την είσοδο *SrcA* των 32 bit της μονάδας ALU
 - σύνδεση της εξόδου *ExtImm* της μονάδας *Extend* με την είσοδο *SrcB* των 32 bit της μονάδας ALU
 - στη μονάδα ALU εκτελείται η πράξη $ALUResult = SrcA + SrcB$
 - σήματα ελέγχου: **ALUSrc = 1** και **ALUControl(2:0) = 000 (+)**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

Ανάγνωση μνήμης δεδομένων

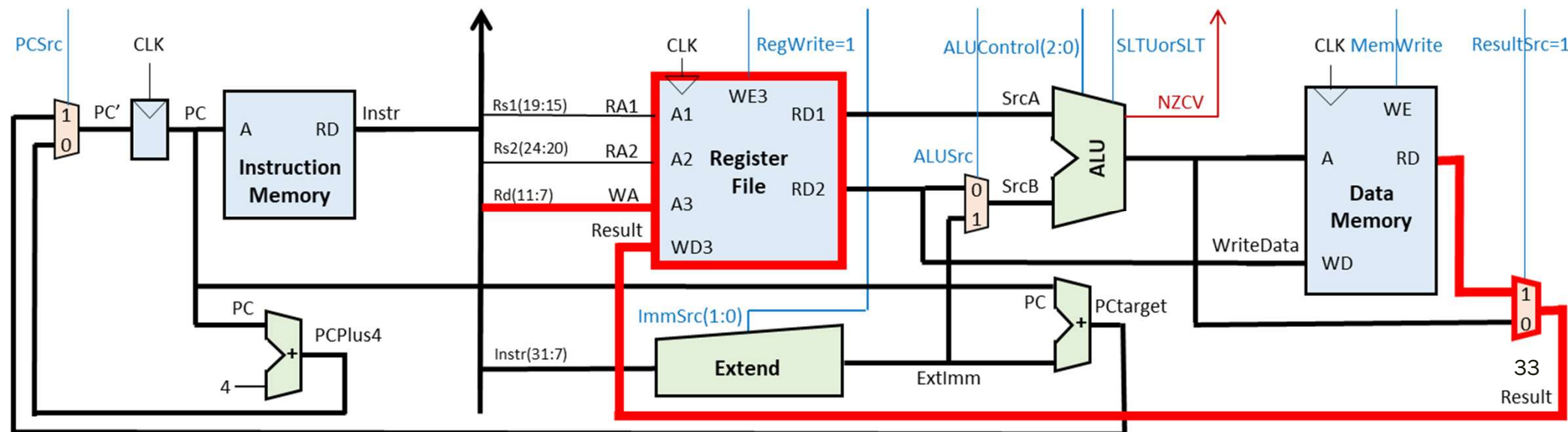
- Η διεύθυνση που εμφανίζεται στην έξοδο ALUResult της μονάδας ALU διευθυνσιοδοτεί την είσοδο διευθύνσεων A της μνήμης δεδομένων (DM)
- Η λέξη δεδομένων διαβάζεται σύγχρονα (Block RAM) ή ασύγχρονα (Distributed RAM) στη θύρα ανάγνωσης RD της DM των $2^M \times 32$ bit
 - η διεύθυνση της λέξης δεδομένων είναι ευθυγραμμισμένη και αφορά μνήμες οργανωμένες σε byte, ενώ η μνήμη δεδομένων DM είναι οργανωμένη σε λέξεις
 - ισχύει $A(M-1:0) = ALUResult(M+1:2)$
 - σήματα ελέγχου: **MemWrite = 0** (διάβασμα μνήμης δεδομένων)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (x0-x31)

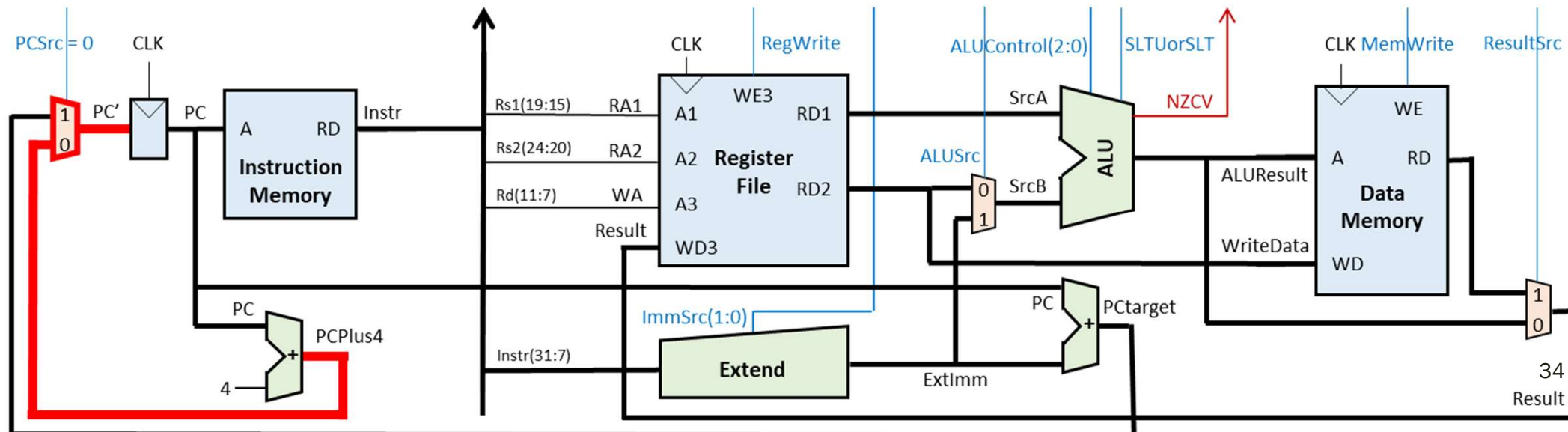
- Η **λέξη δεδομένων** που διαβάζεται από τη μνήμη δεδομένων εγγράφεται ετεροχρονισμένα στον **καταχωρητή προορισμού Rd** του αρχείου καταχωρητών
 - σύνδεση του πεδίου *Rd* της εντολής (*Instr11:7*) στην είσοδο διευθύνσεων εγγραφής *A3*
 - σύνδεση της θύρας ανάγνωσης *RD* της μνήμης δεδομένων με τη θύρα εγγραφής *WD3* του αρχείου καταχωρητών μέσω του πολυπλέκτη *Result*
 - σύγχρονη εγγραφή περιεχομένου καταχωρητή από τη θύρα εγγραφής *WD3*
 - σήματα ελέγχου: **ResultSrc = 1** και **RegWrite = 1** (εγγραφή αρχείου καταχωρητών)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

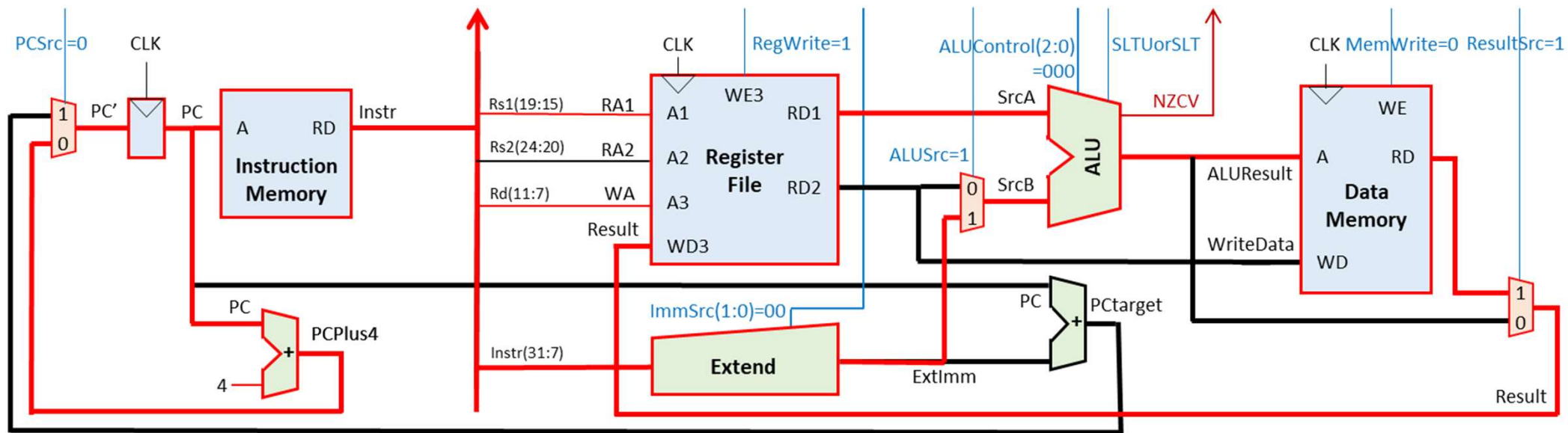
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει την αμέσως επόμενη εντολή, $PC' = PC + 4$
 - σύνδεση της εισόδου 0 του πολυπλέκτη PCSrc με την έξοδο PCPlus4 του αθροιστή κατά 4
 - σύνδεση της εξόδου του πολυπλέκτη PCSrc με την είσοδο PC' του μετρητή PC
 - σήματα ελέγχου: $PCSrc = 0$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

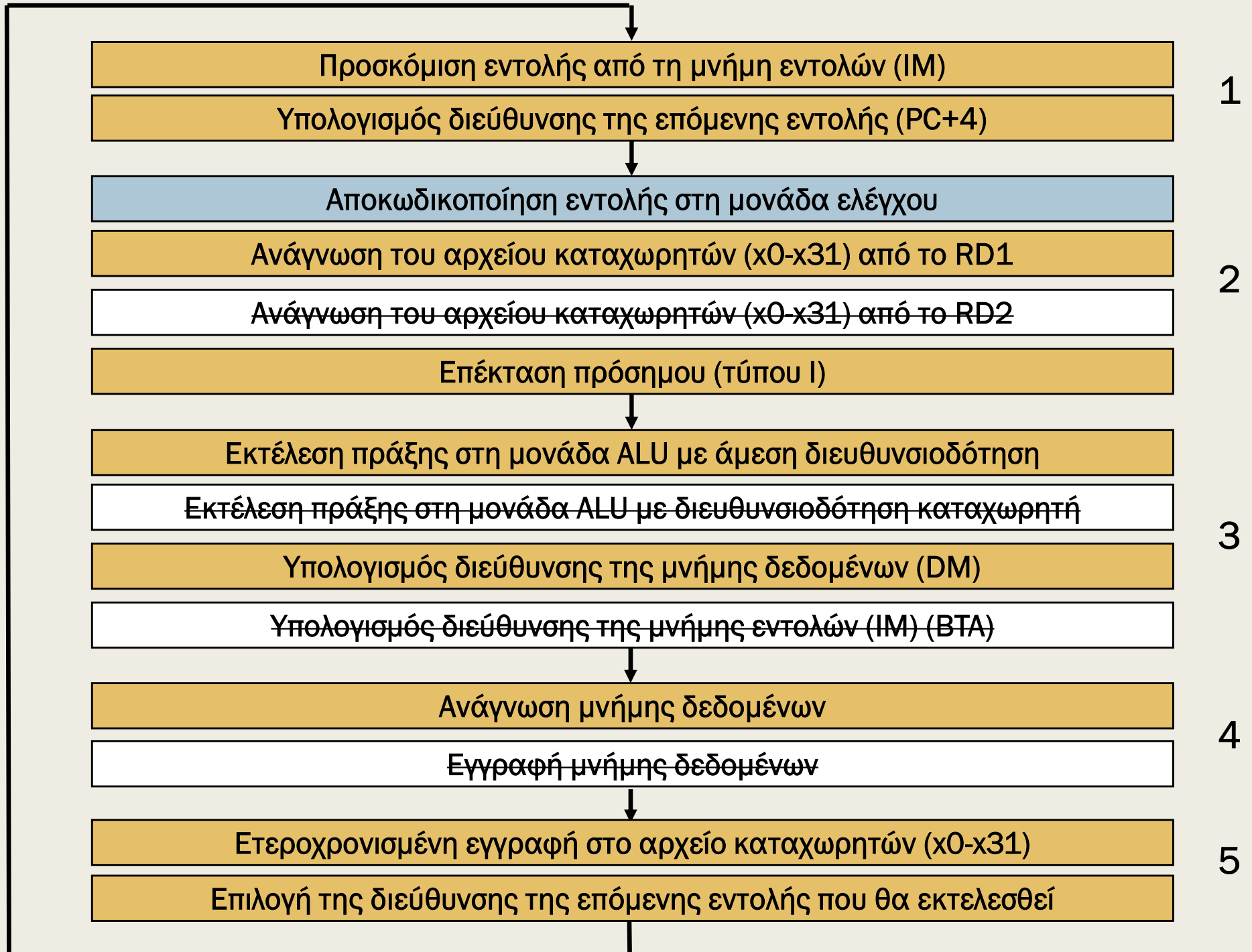


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

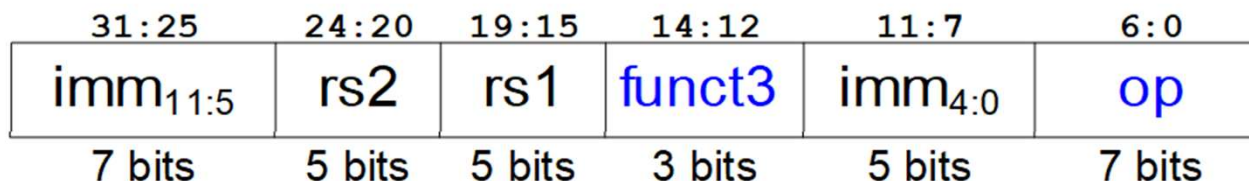


Λειτουργίες εντολής LW (10)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

- Στη συνέχεια, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί την εντολή SW
- Κώδικας συμβολικής γλώσσας:
 - `SW Rs2, imm(Rs1) mem[Rs1+S_Ext(imm)] = Rs2`
- Στη μορφή της εντολής ορίζονται:
 - οι διευθύνσεις του *πρώτου καταχωρητή προέλευσης Rs1* και του *δεύτερου καταχωρητή προέλευσης Rs2*
 - ένας *προσημασμένος άμεσος τελεστέος imm(11:0)* των *12 bit* ($instr(31:25) \& instr(11:7)$)
- Απαιτείται επέκταση προσήμου τύπου S του άμεσου τελεστέου από τα 12 bit στα 32 bit για τον υπολογισμό της σχετικής απόστασης
- Απαιτείται η εκτέλεση της πράξης της *πρόσθεσης* στη μονάδα ALU για τον προσδιορισμό της διεύθυνσης της μνήμης δεδομένων



S-Type

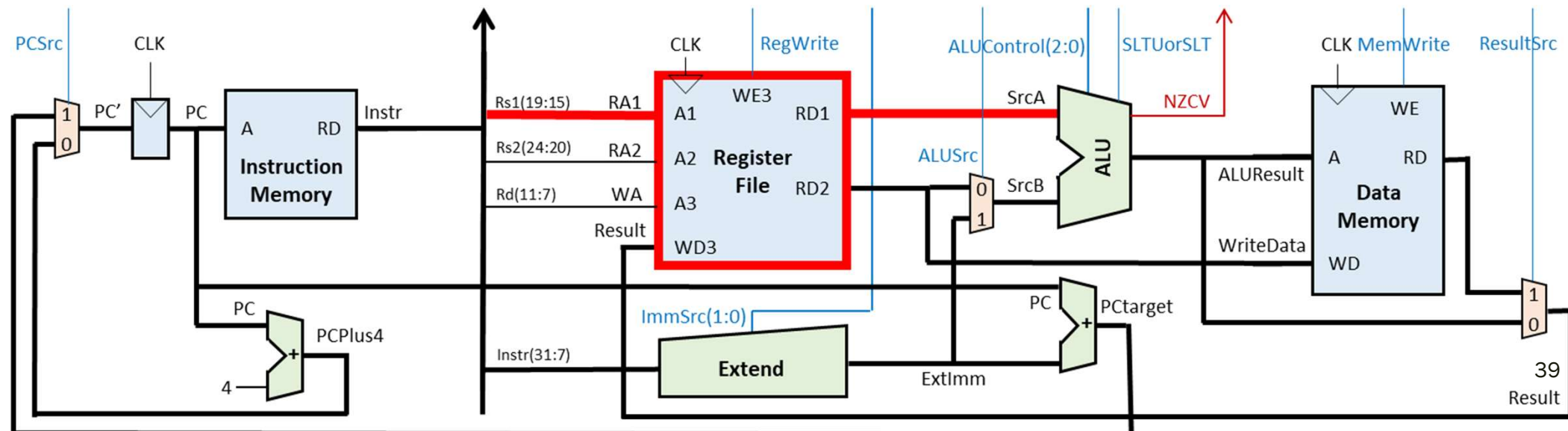
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

- Απαιτούμενες λειτουργίες της εντολής SW που ήδη υποστηρίζονται:
 - η διεύθυνση βάσης που είναι αποθηκευμένη στον πρώτο καταχωρητή προέλευσης $Rs1$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD1$
 - η σχετική απόσταση προστίθεται στη διεύθυνση βάσης, ώστε να προκύψει η διεύθυνση της μνήμης δεδομένων
 - ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση της αμέσως επόμενης εντολής: $PC' = PC + 4$
- Νέες λειτουργίες για την εντολή SW:
 - η λέξη δεδομένων που είναι αποθηκευμένη στον δεύτερο καταχωρητή προέλευσης $Rs2$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD2$
 - επέκταση προσήμου τύπου S από τα 12 bit στα 32 bit για τον υπολογισμό της σχετικής απόστασης
 - σύγχρονη εγγραφή της λέξης δεδομένων στη μνήμη δεδομένων
 - δεν υπάρχει εγγραφή στο αρχείο καταχωρητών

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

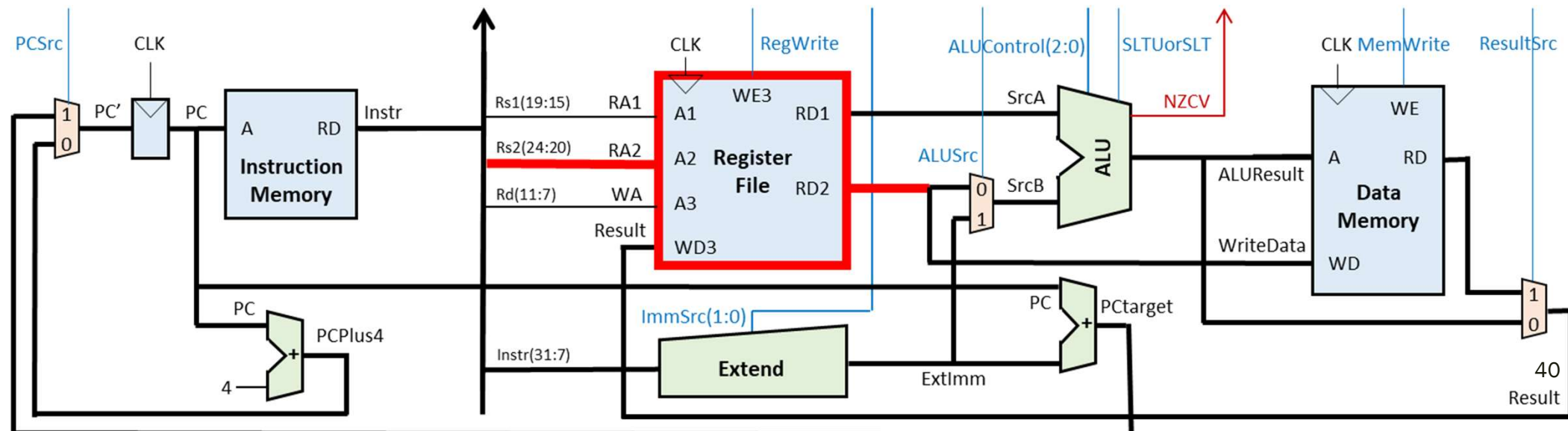
- Η διεύθυνση βάσης που είναι αποθηκευμένη στον **καταχωρητή προέλευσης Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

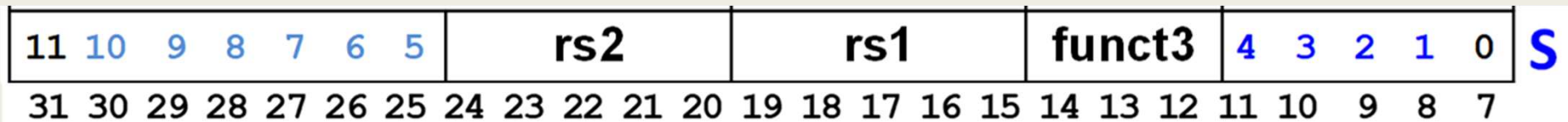
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

- Το περιεχόμενο του **καταχωρητή προέλευσης Rs2** του αρχείου καταχωρητών (x0-x31) που πρόκειται να εγγραφεί διαβάζεται στη θύρα ανάγνωσης RD2
 - σύνδεση του πεδίου *Rs2* της εντολής (*Instr24:20*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD2*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)

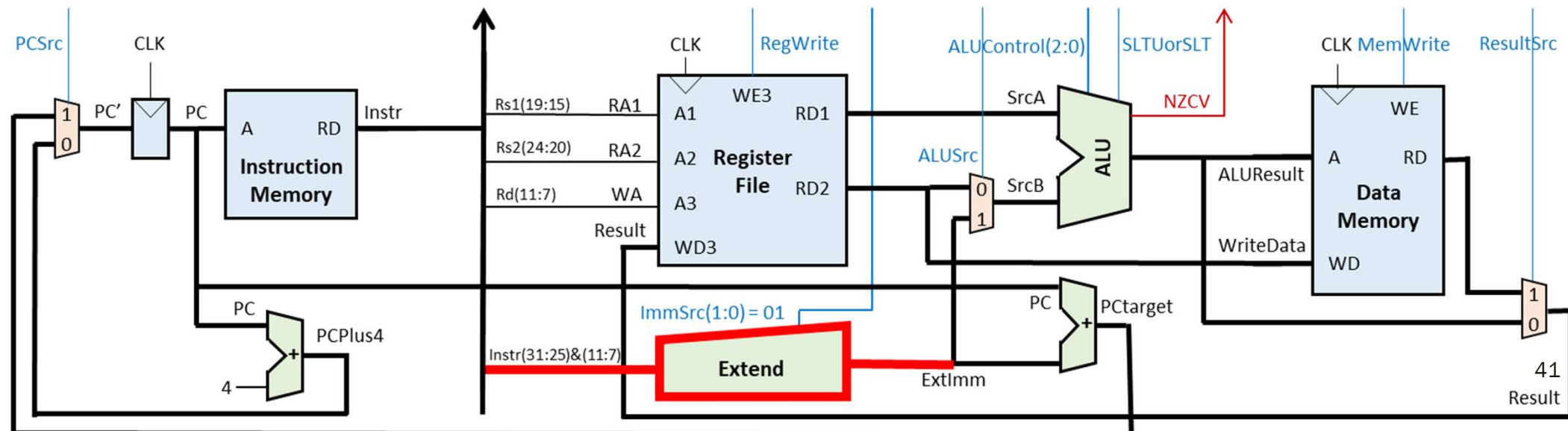


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

Επέκταση πρόσημου (τύπων I, S, B, U) – Επιλογή τύπου S



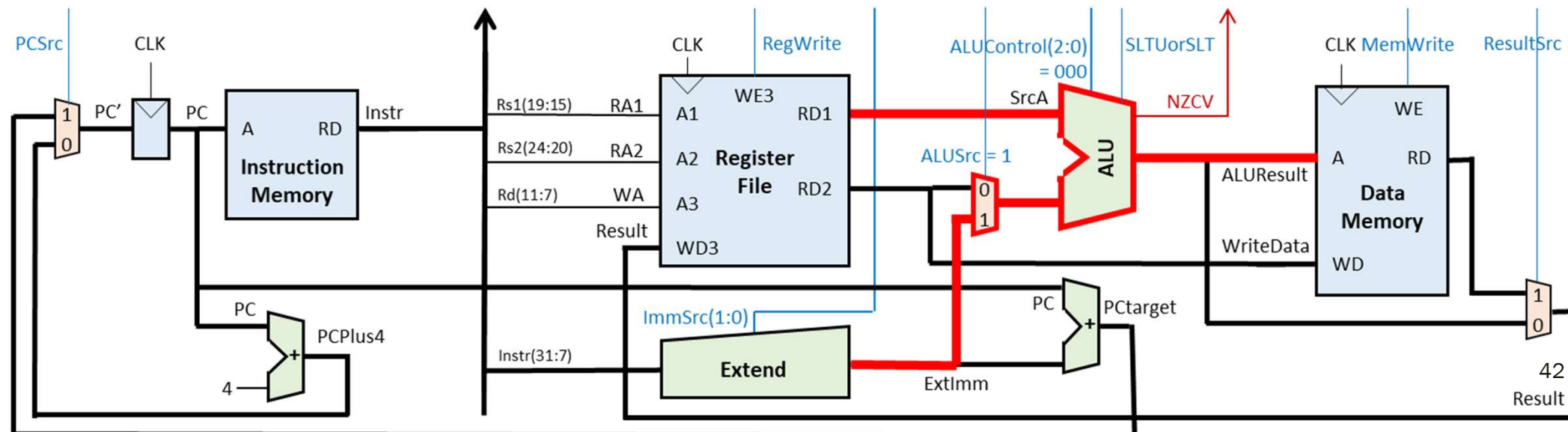
- Η εντολή SW απαιτεί και μια **σχετική απόσταση (offset)** για τον υπολογισμό της διεύθυνσης της μνήμης δεδομένων που προκύπτει με επέκταση προσήμου από τα 12 bit στα 32 bit στην μονάδα Extend
 - $\text{Extend}(31:0) = \text{Sign_Ext}(\text{instr}(31:25) \& \text{instr}(11:7))$
 - Για επέκταση προσήμου τύπου S τα σήματα ελέγχου είναι: **ImmSrc(1:0) = 01**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

Υπολογισμός διεύθυνσης της μνήμης δεδομένων (DM)

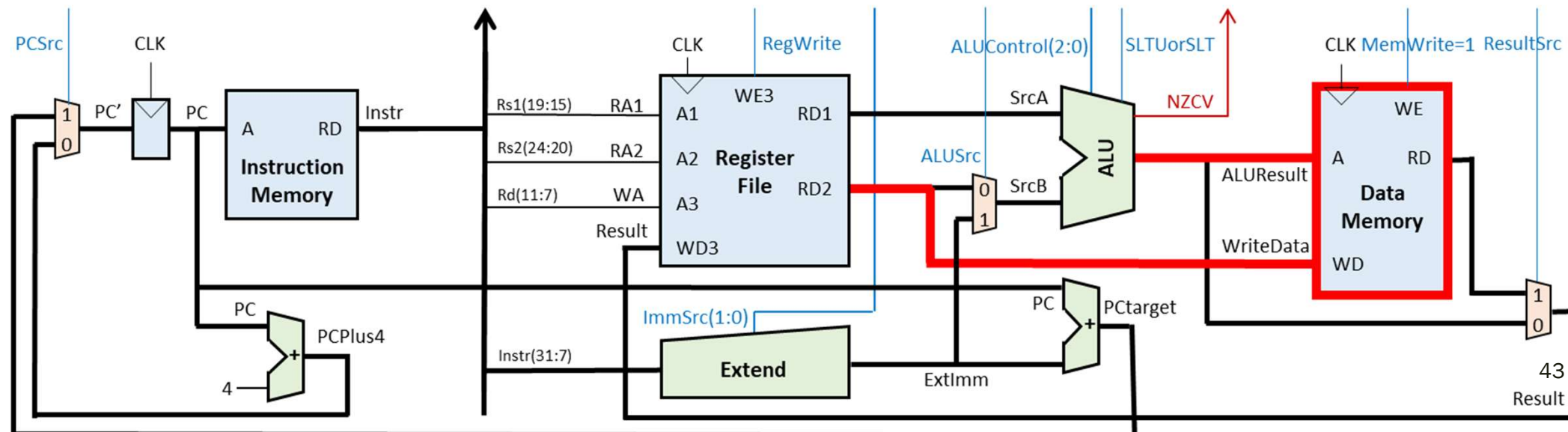
- Η **σχετική απόσταση προστίθεται** στη διεύθυνση βάσης στη μονάδα ALU, ώστε να προκύψει η διεύθυνση της μνήμης δεδομένων ως ALUResult
 - σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA των 32 bit της μονάδας ALU
 - σύνδεση της εξόδου ExtImm της μονάδας Extend με την είσοδο SrcB των 32 bit της μονάδας ALU
 - στη μονάδα ALU εκτελείται η πράξη $ALUResult = SrcA + SrcB$
 - σήματα ελέγχου: **ALUSrc = 1** και **ALUControl(2:0) = 000 (+)**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

Εγγραφή μνήμης δεδομένων

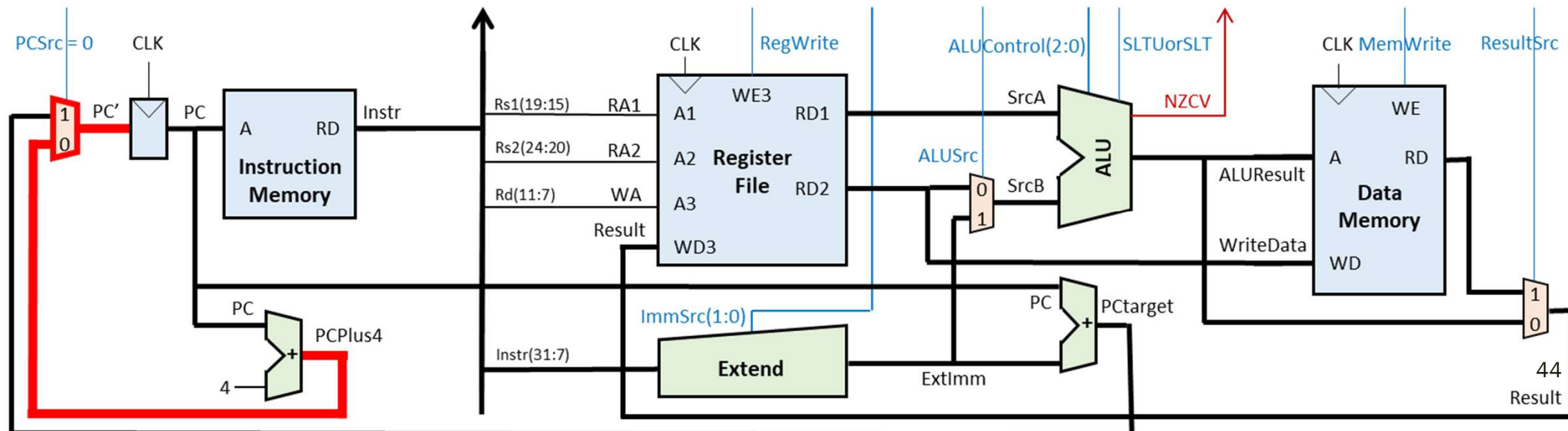
- Η διεύθυνση που εμφανίζεται στην έξοδο ALUResult της μονάδας ALU διευθυνσιοδοτεί την είσοδο διευθύνσεων A της μνήμης δεδομένων (DM)
- Η **λέξη δεδομένων** (WriteData), που εμφανίζεται στη θύρα ανάγνωσης RD2, γράφεται σύγχρονα στη μνήμη δεδομένων DM των $2^M \times 32$ bit μέσω της θύρας εγγραφής δεδομένων WD της DM
 - η διεύθυνση της λέξης δεδομένων είναι ευθυγραμμισμένη και αφορά μνήμες οργανωμένες σε *byte*, ενώ η μνήμη δεδομένων DM είναι οργανωμένη σε λέξεις
 - ισχύει $A(M-1:0) = ALUResult(M+1:2)$
 - σήματα ελέγχου: **MemWrite = 1** (εγγραφή μνήμης δεδομένων)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

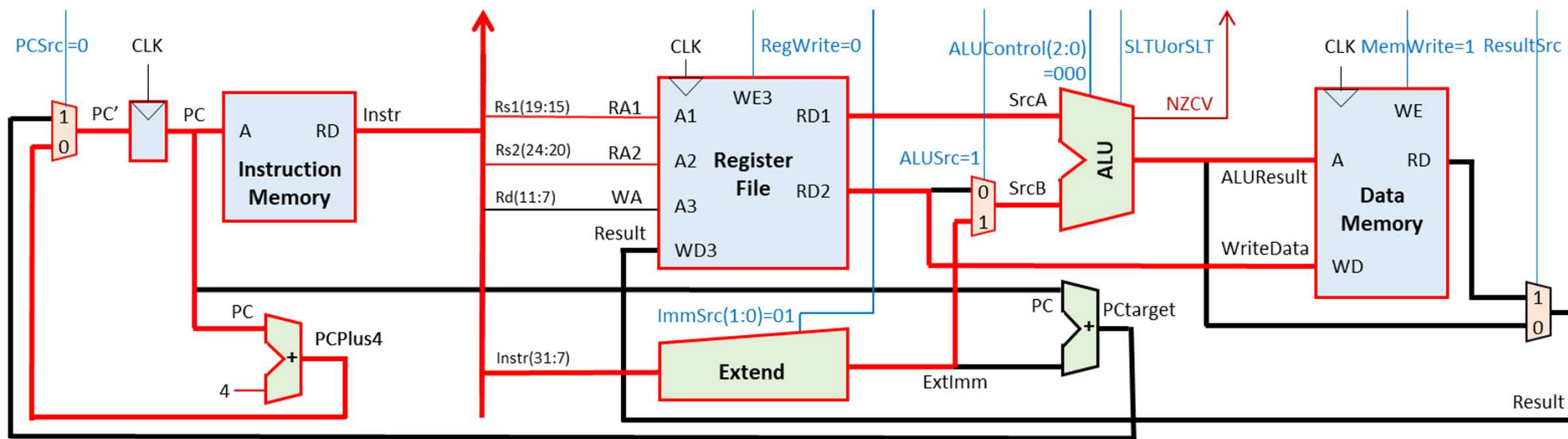
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει την αμέσως επόμενη εντολή, $PC' = PC + 4$
 - σύνδεση της εισόδου 0 του πολυπλέκτη PCSrc με την έξοδο PCPlus4 του αθροιστή κατά 4
 - σύνδεση της εξόδου του πολυπλέκτη PCSrc με την είσοδο PC' του μετρητή PC
 - σήματα ελέγχου: $PCSrc = 0$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

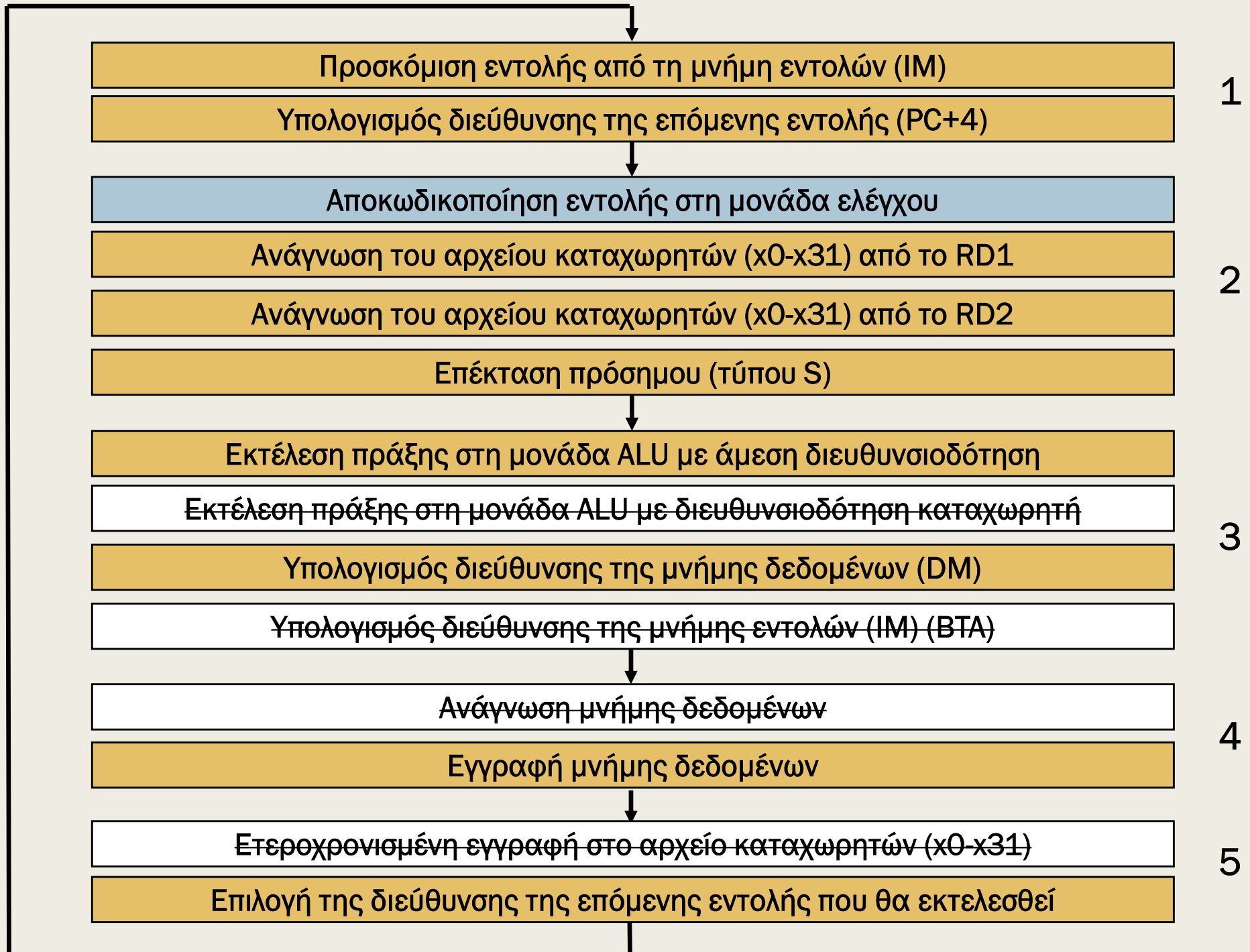


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



Λειτουργίες εντολής SW (10)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

- Στη συνέχεια, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί τις **εντολές εκτέλεσης αριθμητικών και λογικών πράξεων (ADD, SUB, AND, OR και XOR)** με **διευθυνσιοδότηση καταχωρητή**
- Κώδικας συμβολικής γλώσσας των εντολών ALU:
 - **ADD** **Rd, Rs1, Rs2** **Rd = Rs1 + Rs2**
 - **SUB** **Rd, Rs1, Rs2** **Rd = Rs1 - Rs2**
 - **AND** **Rd, Rs1, Rs2** **Rd = Rs1 and Rs2**
 - **OR** **Rd, Rs1, Rs2** **Rd = Rs1 or Rs2**
 - **XOR** **Rd, Rs1, Rs2** **Rd = Rs1 xor Rs2**
- Στη μορφή των εντολών ορίζονται:
 - οι διευθύνσεις των δύο **καταχωρητών προέλευσης Rs1 και Rs2**, καθώς και του **καταχωρητή προορισμού Rd**

31:25	24:20	19:15	14:12	11:7	6:0
funct7	rs2	rs1	funct3	rd	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

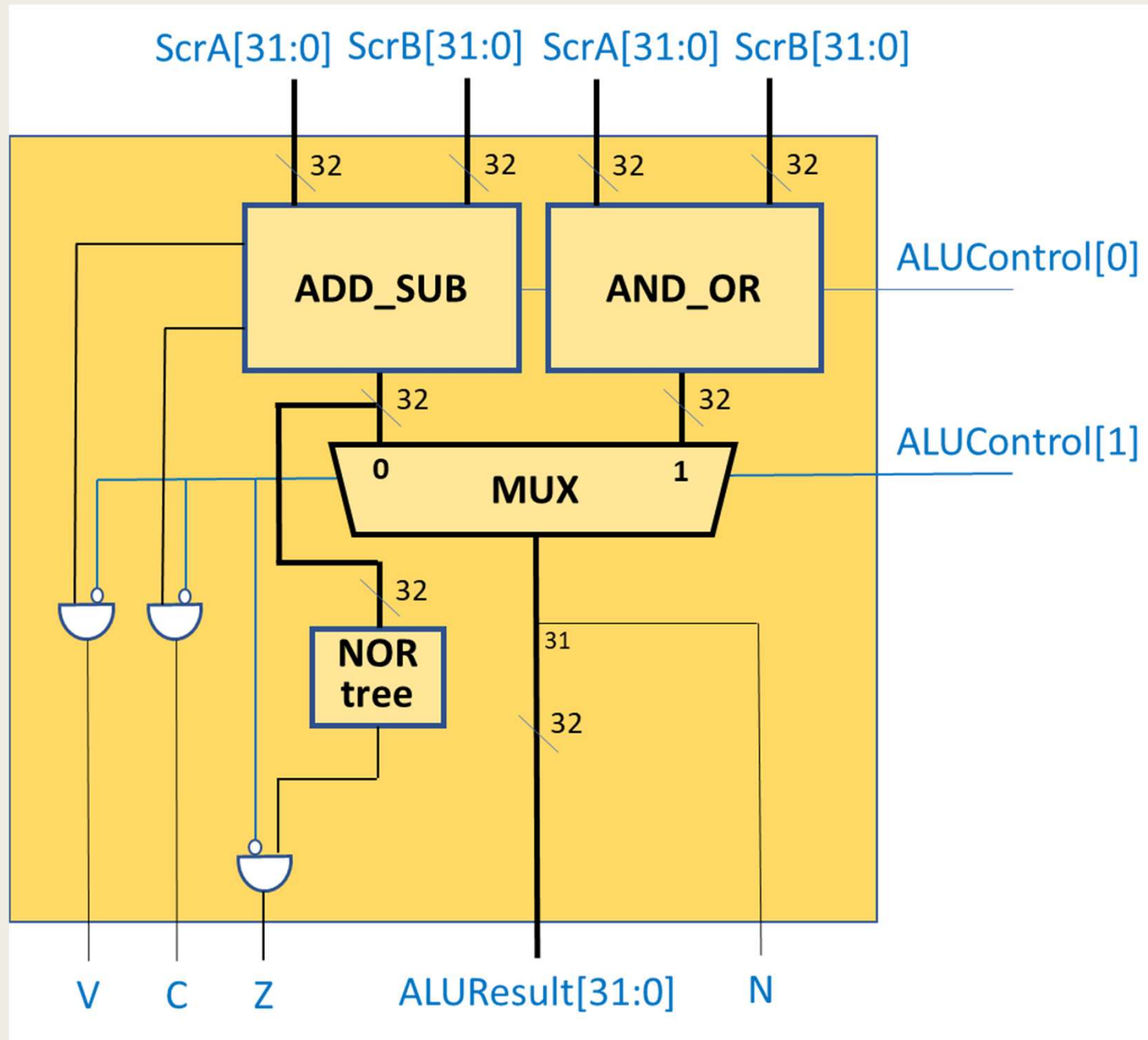
- Απαιτούμενες λειτουργίες των εντολών ALU που ήδη υποστηρίζονται:
 - ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs1$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD1$
 - ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs2$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD2$
 - Το αποτέλεσμα των πράξεων στη μονάδα ALU (ALUResult) γράφεται ετερο-χρονισμένα στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
 - ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση της αμέσως επόμενης εντολής: $PC' = PC + 4$
- Νέες λειτουργίες για τις εντολές ALU:
 - η μονάδα ALU πρέπει να υποστηρίζει τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR
 - εκτελούνται οι αριθμητικές και λογικές πράξεις με διευθυνσιοδότηση καταχωρητή

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Η μονάδα ALU

- Η μονάδα ALU πρέπει να υλοποιεί:
 - τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, και
 - τις λογικές πράξεις AND, OR και XOR
- Η μονάδα ALU λαμβάνει το σήμα ελέγχου **ALUControl(2:0)** που ορίζει τη λειτουργία της και τις εισόδους **SrcA** και **SrcB** των 32 bit
- Η μονάδα ALU παράγει το αποτέλεσμα της πράξης **ALUResult** των 32 bit και τις **σημείες N, Z, C, V** (Z = 0, C = 0 και V = 0 στις λογικές πράξεις)

ALUControl(2:0)	Πράξη
000	Add
001	Subtract
011 (SLTUorSLT=0)	SLT
011 (SLTUorSLT=1)	SLTU
100	AND
101	OR
11X	XOR

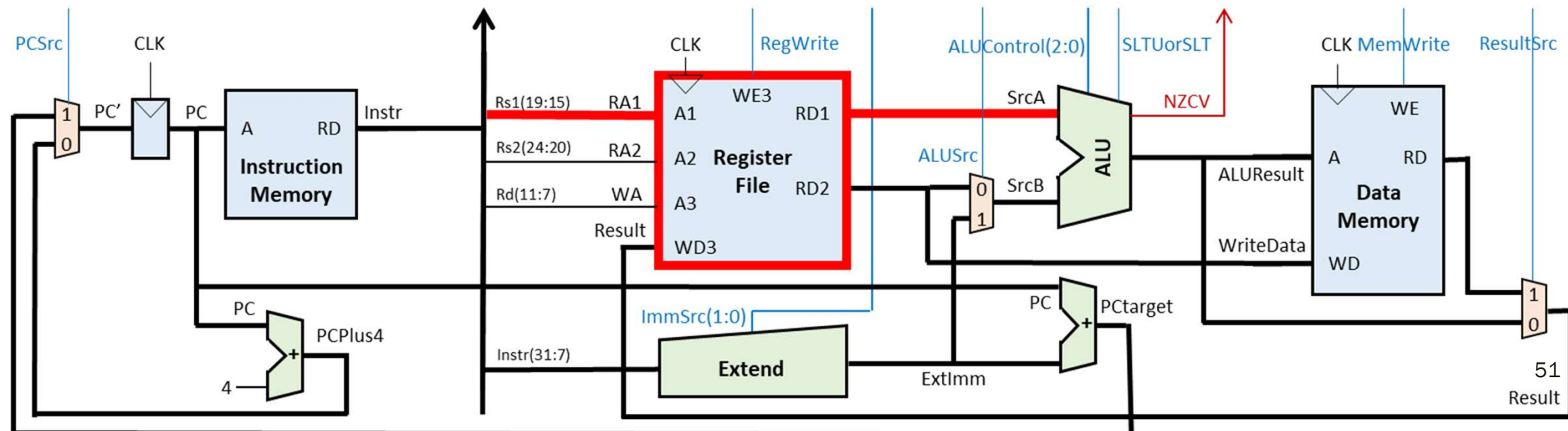
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Η μονάδα ALU



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

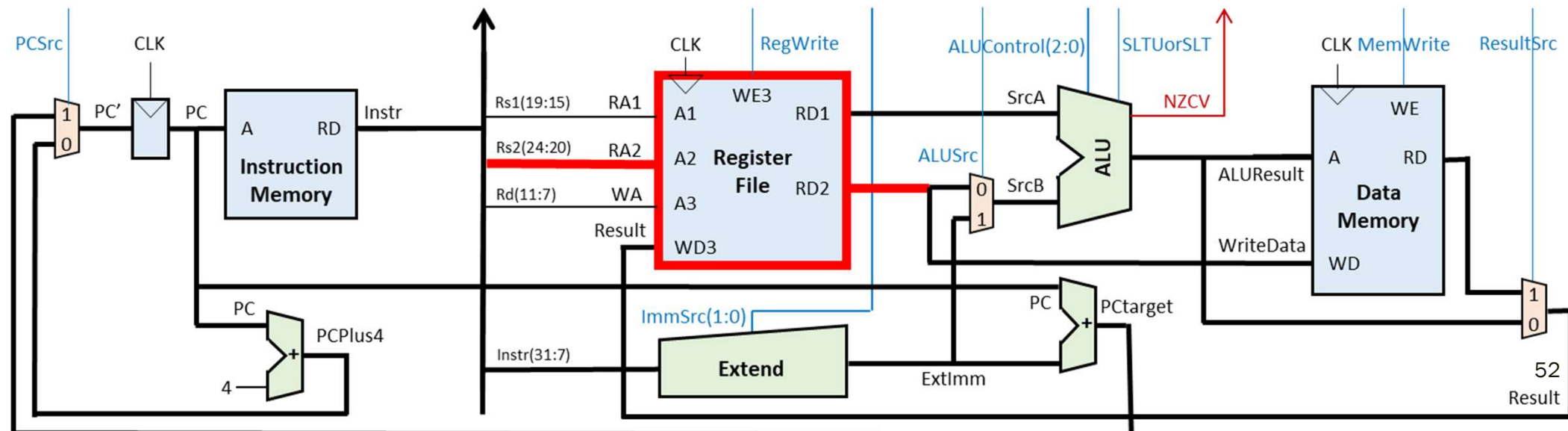
- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

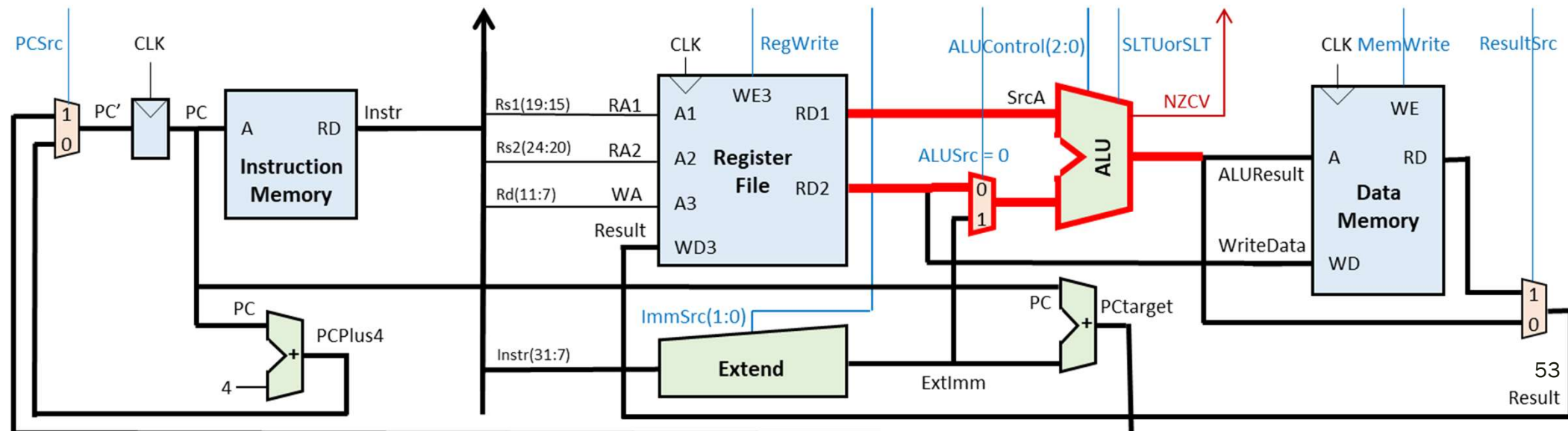
- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs2** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD2
 - σύνδεση του πεδίου *Rs2* της εντολής (*Instr24:20*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD2*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

Εκτέλεση πράξης στη μονάδα ALU με διευθυνσιοδότηση καταχωρητή

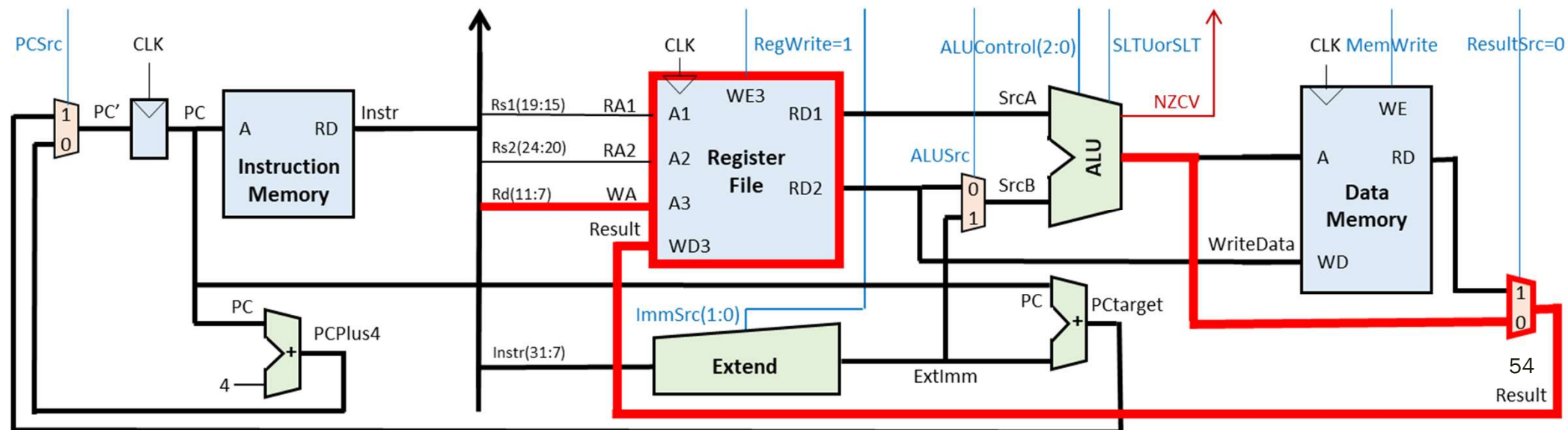
- Η μονάδα ALU εκτελεί τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR μετά από:
 - σύνδεση της θύρας *RD1* του αρχείου καταχωρητών με την είσοδο *SrcA* των 32 bit της μονάδας ALU
 - σύνδεση της θύρας *RD2* του αρχείου καταχωρητών με την είσοδο *SrcB* των 32 bit της μονάδας ALU
- Εκτελείται αριθμητική ή λογική πράξη και το αποτέλεσμα της πράξης εμφανίζεται στην έξοδο *ALUResult* των 32 bit της μονάδας ALU
- Σήματα ελέγχου: **ALUSrc = 0** και **ALUControl(2:0)** σύμφωνα με τον πίνακα



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (x0-x31)

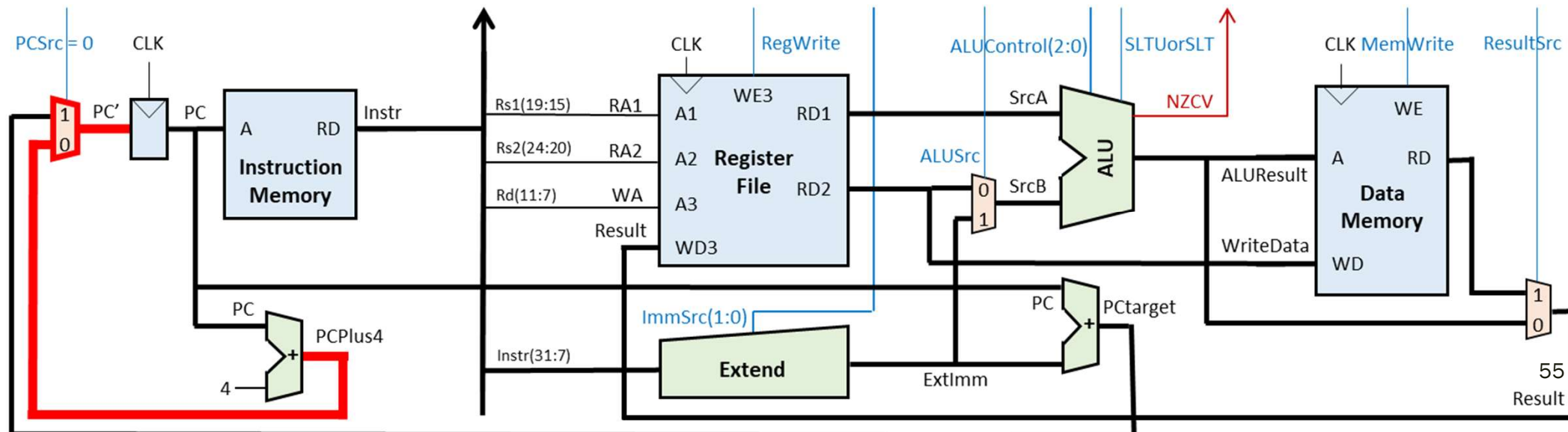
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον **καταχωρητή προορισμού Rd** του αρχείου καταχωρητών
 - σύνδεση του πεδίου *Rd* της εντολής (*Instr11:7*) στην είσοδο διευθύνσεων εγγραφής *A3*
 - σύνδεση της εξόδου *ALUResult* της μονάδας ALU με τη θύρα εγγραφής *WD3* του αρχείου καταχωρητών μέσω του πολυπλέκτη *Result*
 - σύγχρονη εγγραφή περιεχομένου καταχωρητή από τη θύρα εγγραφής *WD3*
 - σήματα ελέγχου: **ResultSrc = 0** και **RegWrite = 1** (εγγραφή αρχείου καταχωρητών)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

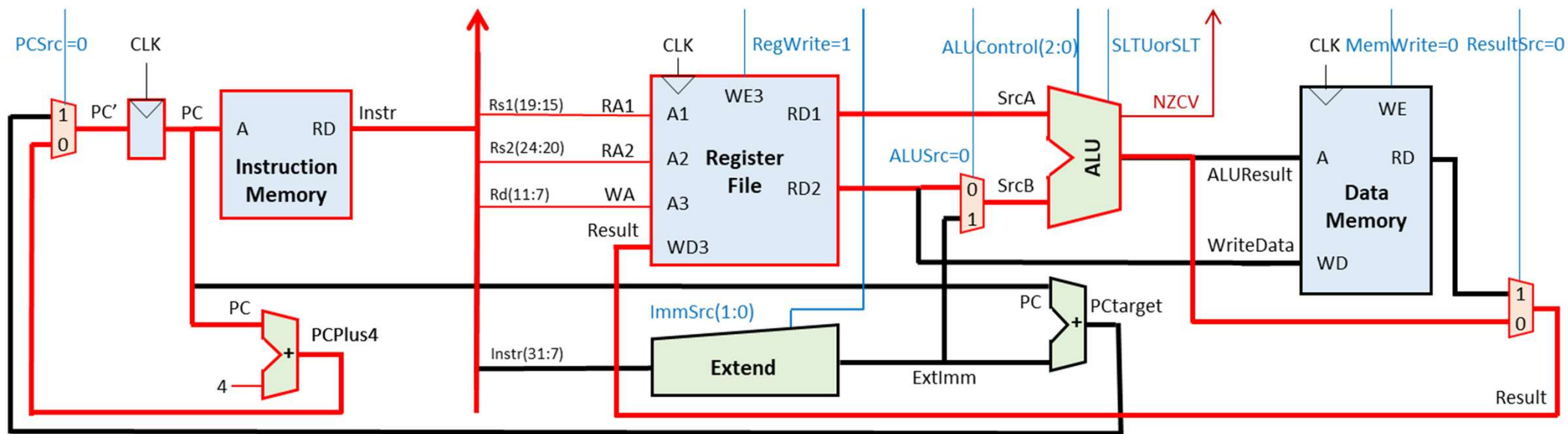
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει την αμέσως επόμενη εντολή, $PC' = PC + 4$
 - σύνδεση της εισόδου 0 του πολυπλέκτη PCSrc με την έξοδο PCPlus4 του αθροιστή κατά 4
 - σύνδεση της εξόδου του πολυπλέκτη PCSrc με την είσοδο PC' του μετρητή PC
 - σήματα ελέγχου: $PCSrc = 0$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

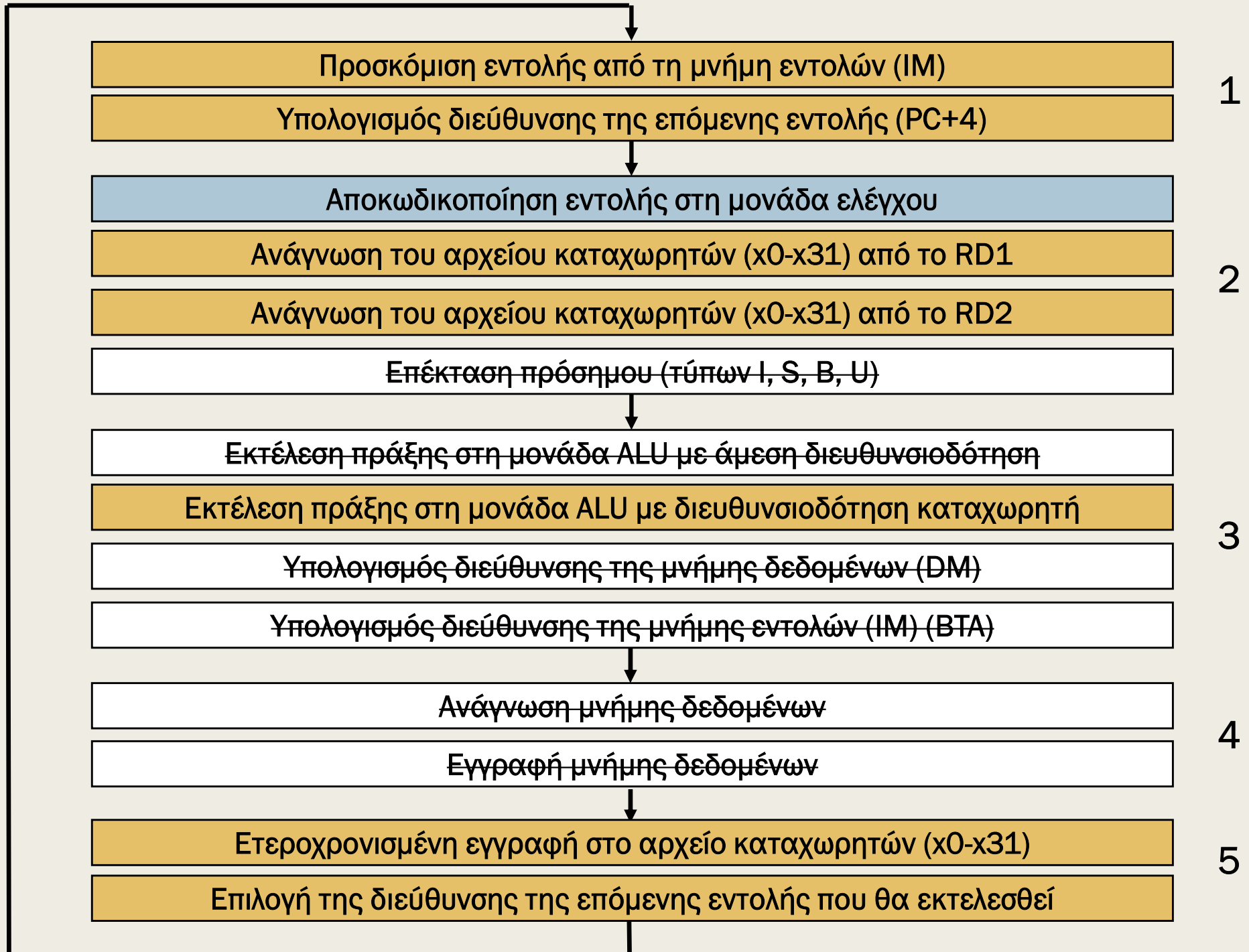


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



Λειτουργίες εντολών ALU (8)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

- Στη συνέχεια, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί τις **εντολές εκτέλεσης αριθμητικών και λογικών πράξεων (ADDI, ANDI, ORI και XORI)** με **άμεση διευθυνσιοδότηση**
- Κώδικας συμβολικής γλώσσας των εντολών ALU:
 - **ADDI** *Rd, Rs1, imm* $Rd = Rs1 + S_Ext(imm)$
 - **ANDI** *Rd, Rs1, imm* $Rd = Rs1 \text{ and } S_Ext(imm)$
 - **ORI** *Rd, Rs1, imm* $Rd = Rs1 \text{ or } S_Ext(imm)$
 - **XORI** *Rd, Rs1, imm* $Rd = Rs1 \text{ xor } S_Ext(imm)$
- Στη μορφή των εντολών ορίζονται:
 - οι διευθύνσεις του **καταχωρητή προέλευσης *Rs1*** και του **καταχωρητή προορισμού *Rd***
 - ένας **προσημασμένος άμεσος τελεστής *imm(11:0)*** των **12 bit** (*instr(31:20)*)
- Απαιτείται επέκταση προσήμου τύπου I του άμεσου τελεστή από τα 12 bit στα 32 bit για τον υπολογισμό του άμεσου τελεστή

31:20	19:15	14:12	11:7	6:0
imm _{11:0}	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

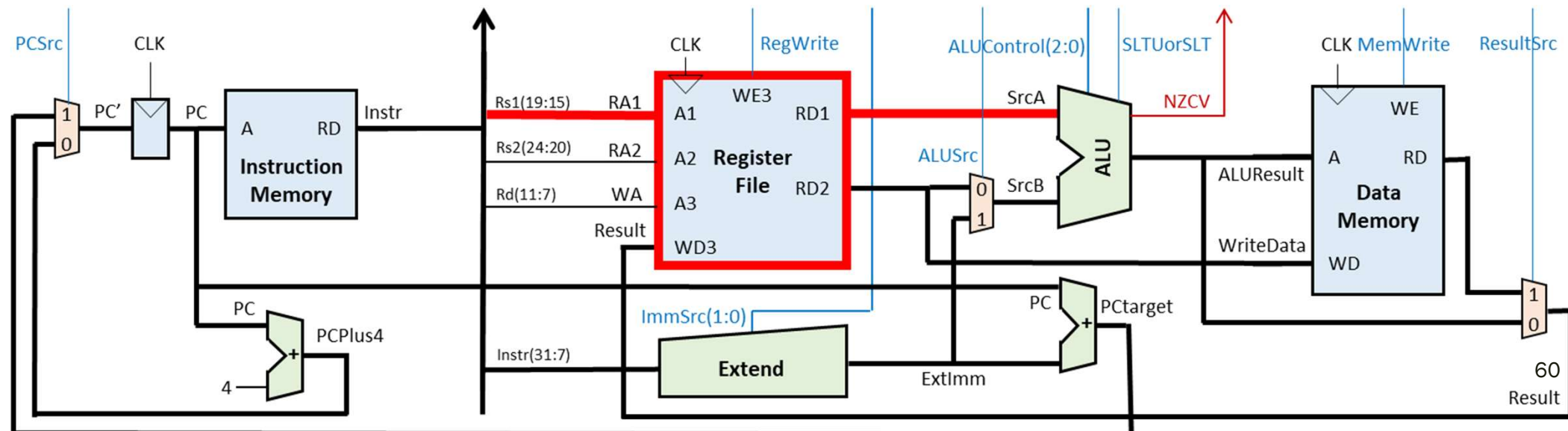
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

- Απαιτούμενες λειτουργίες των εντολών ALUI που ήδη υποστηρίζονται:
 - ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs1$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD1$
 - επέκταση προσήμου τύπου I από τα 12 bit στα 32 bit για τον υπολογισμό του άμεσου τελεστέου
 - η μονάδα ALU πρέπει να υποστηρίζει τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR
 - Το αποτέλεσμα των πράξεων στη μονάδα ALU (ALUResult) γράφεται ετεροχρονισμένα στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
 - ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση της αμέσως επόμενης εντολής: $PC' = PC + 4$
- Νέες λειτουργίες για τις εντολές ALUI:
 - εκτελούνται οι αριθμητικές και λογικές πράξεις με άμεση διευθυνσιοδότηση

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

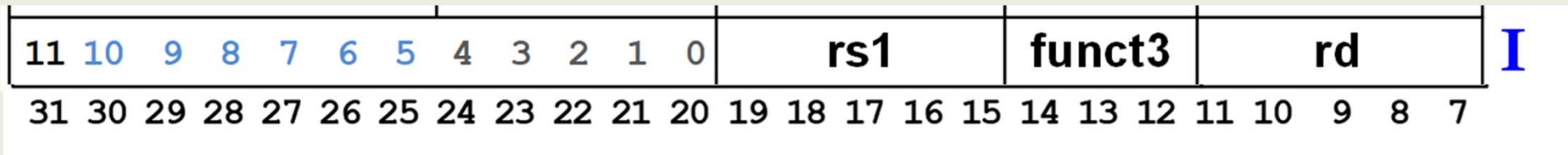
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)

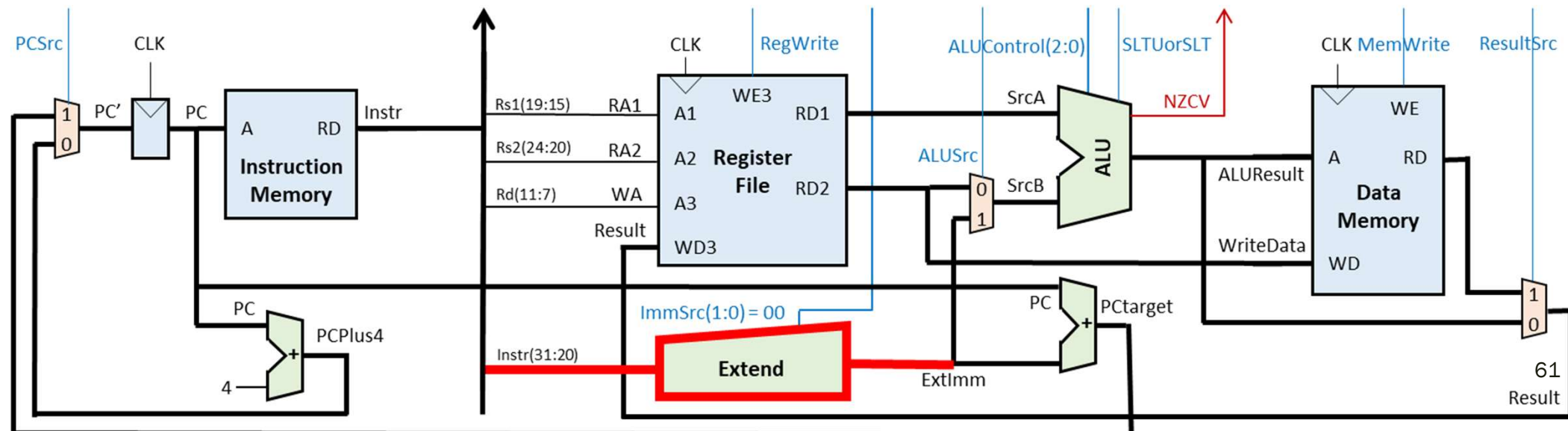


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

Επέκταση πρόσημου (τύπων I, S, B, U) – Επιλογή τύπου I



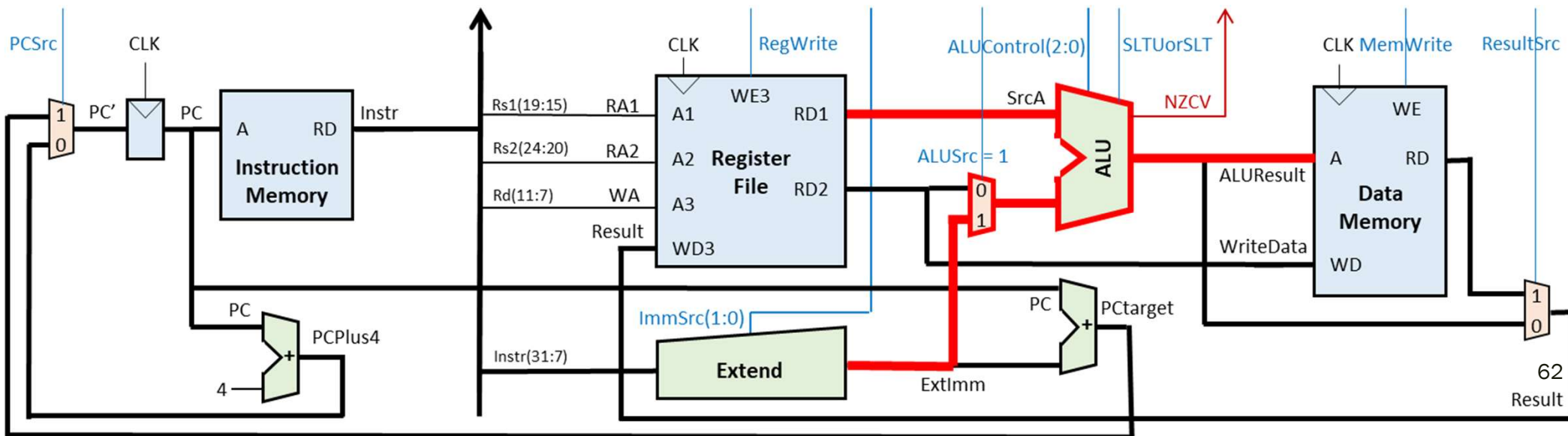
- Η εντολές ALUI διαθέτουν έναν **προσημασμένο άμεσο τελεστέο imm(11:0)** των **12 bit** (instr(31:20)) που για να χρησιμοποιηθεί στις πράξεις πρέπει πρώτα να υποστεί επέκταση πρόσημου από τα 12 bit στα 32 bit στην μονάδα Extend
 - $\text{Extend}(31:0) = \text{Sign_Ext}(\text{instr}(31:20))$
 - Για επέκταση πρόσημου τύπου I τα σήματα ελέγχου είναι: **ImmSrc(1:0) = 00**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

Εκτέλεση πράξης στη μονάδα ALU με άμεση διευθυνσιοδότηση

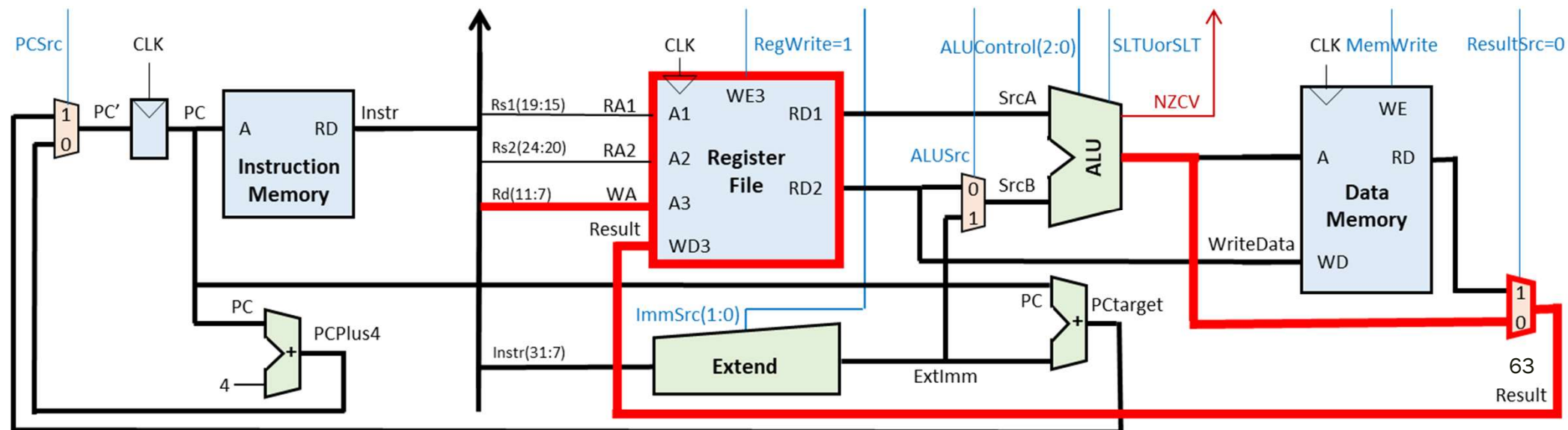
- Η μονάδα ALU εκτελεί τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR τύπου I μετά από:
 - σύνδεση της θύρας *RD1* του αρχείου καταχωρητών με την είσοδο *SrcA* των 32 bit της μονάδας ALU
 - σύνδεση της εξόδου *ExtImm* της μονάδας *Extend* με την είσοδο *SrcB* των 32 bit της μονάδας ALU
- Εκτελείται αριθμητική ή λογική πράξη και το αποτέλεσμα της πράξης εμφανίζεται στην έξοδο *ALUResult* των 32 bit της μονάδας ALU
- Σήματα ελέγχου: **ALUSrc = 1** και **ALUControl(2:0)** σύμφωνα με τον πίνακα



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (x0-x31)

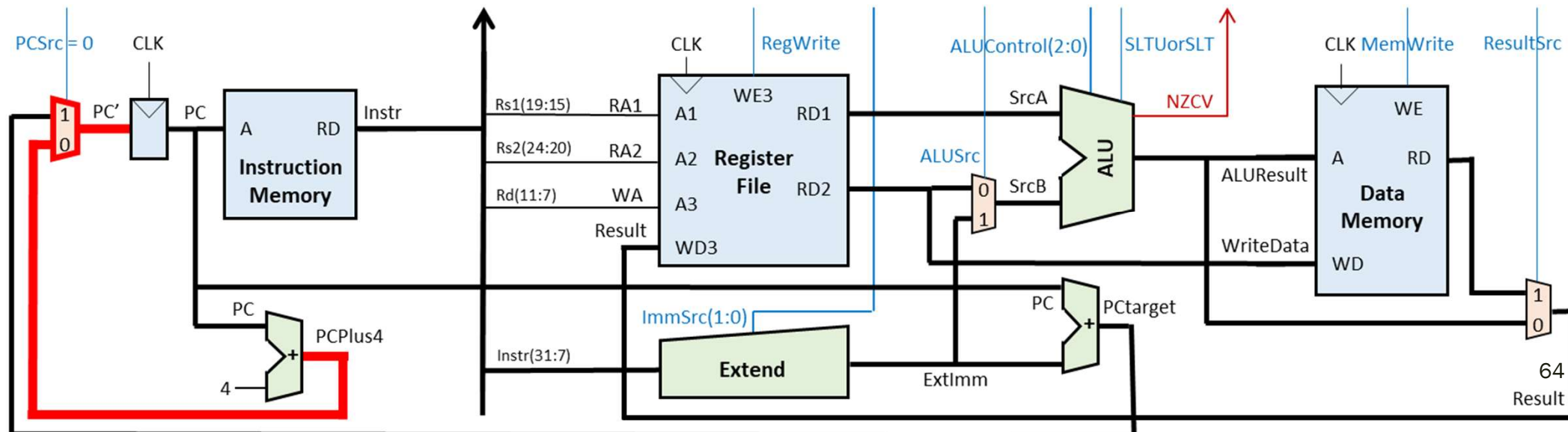
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον **καταχωρητή προορισμού Rd** του αρχείου καταχωρητών
 - σύνδεση του πεδίου *Rd* της εντολής (*Instr11:7*) στην είσοδο διευθύνσεων εγγραφής *A3*
 - σύνδεση της εξόδου *ALUResult* της μονάδας *ALU* με τη θύρα εγγραφής *WD3* του αρχείου καταχωρητών μέσω του πολυπλέκτη *Result*
 - σύγχρονη εγγραφή περιεχομένου καταχωρητή από τη θύρα εγγραφής *WD3*
 - σήματα ελέγχου: **ResultSrc = 0** και **RegWrite = 1** (εγγραφή αρχείου καταχωρητών)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

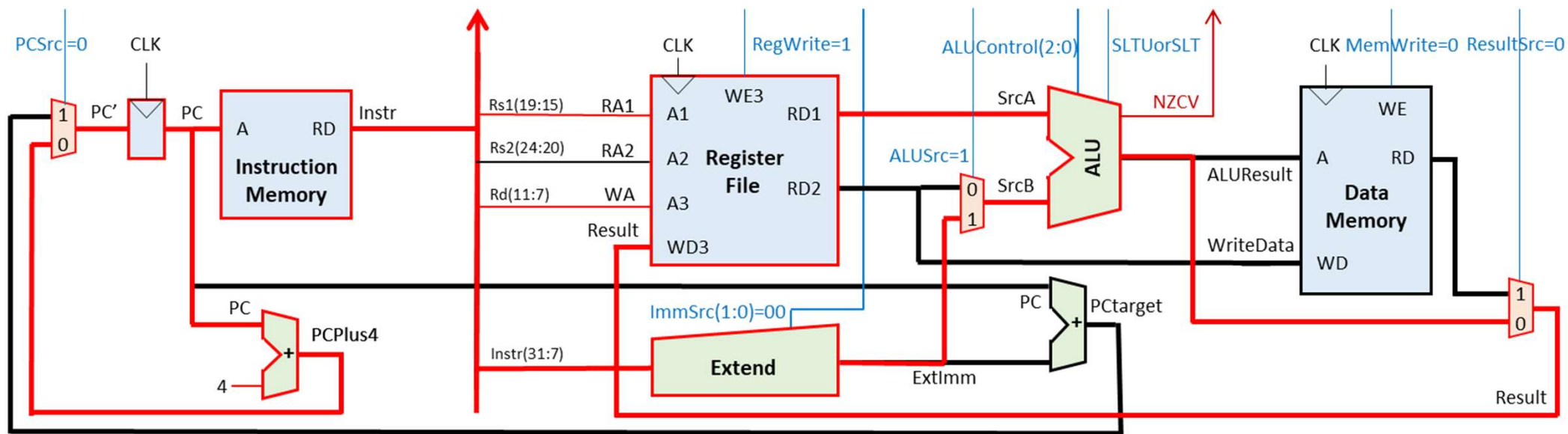
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει την αμέσως επόμενη εντολή, $PC' = PC + 4$
 - σύνδεση της εισόδου 0 του πολυπλέκτη $PCSrc$ με την έξοδο $PCPlus4$ του αθροιστή κατά 4
 - σύνδεση της εξόδου του πολυπλέκτη $PCSrc$ με την είσοδο PC' του μετρητή PC
 - σήματα ελέγχου: $PCSrc = 0$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

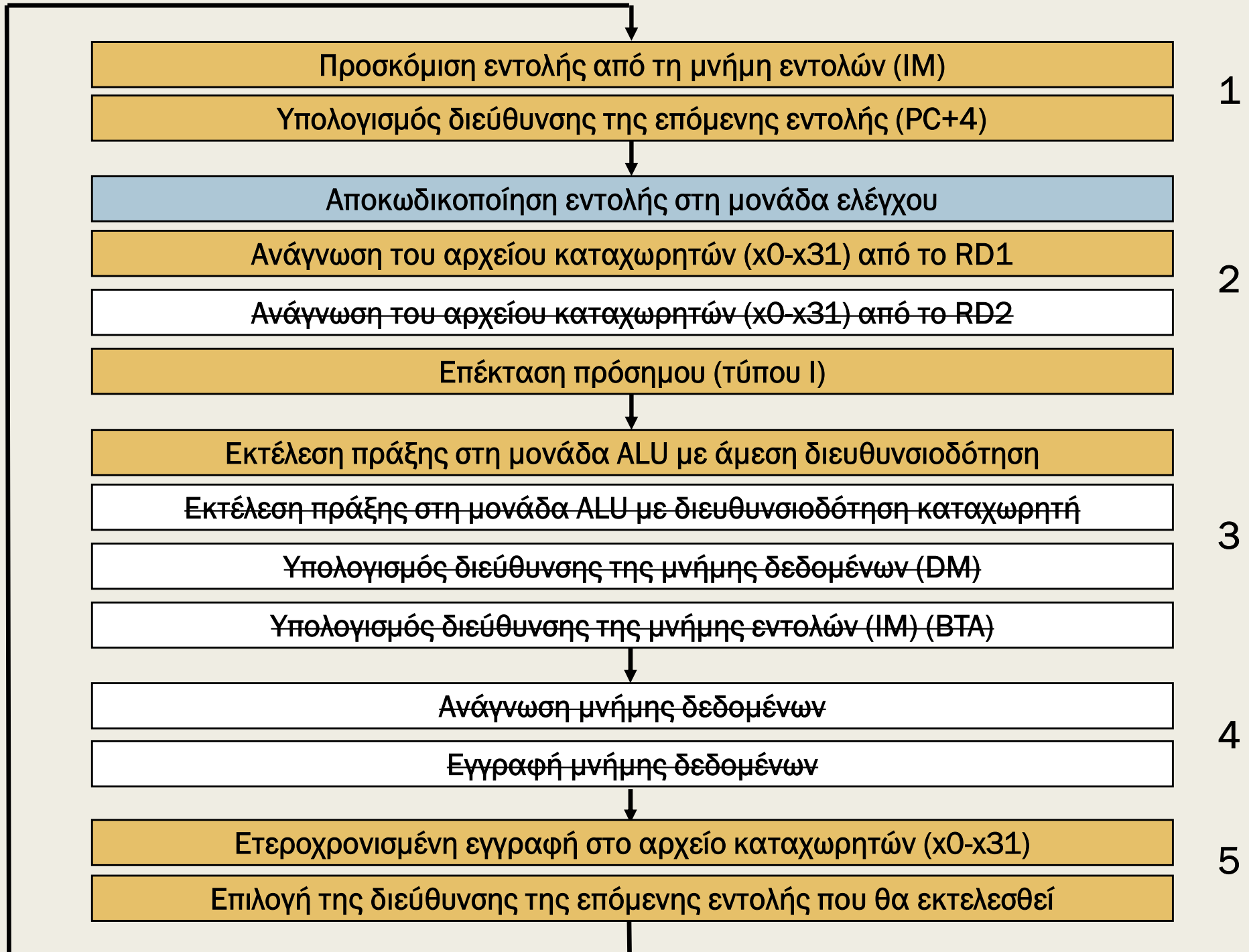


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



Λειτουργίες εντολών ALUI (8)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

- Στη συνέχεια, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί τις **εντολές εκτέλεσης SLT** (για προσημασμένη σύγκριση) και **SLTU** (για μη προσημασμένη σύγκριση) με **διευθυνσιοδότηση καταχωρητή**
- Κώδικας συμβολικής γλώσσας των εντολών SLT/SLTU:
 - **SLT** Rd, Rs1, Rs2 Rd = 1 if rs1<rs2, else 0
 - **SLTU** Rd, Rs1, Rs2 Rd = 1 if rs1<rs2, else 0
- Στη μορφή των εντολών ορίζονται:
 - οι διευθύνσεις των δύο **καταχωρητών προέλευσης Rs1** και **Rs2**, καθώς και του **καταχωρητή προορισμού Rd**
- Η σύγκριση υλοποιείται με αφαίρεση (Rs1 – Rs2) ώστε να παραχθούν τα απαραίτητα ALU flags N,C,V
 - **SLT:** Rd = 1 if $N \oplus V = 1$, else 0 ($N \oplus V = 0$)
 - **SLTU:** Rd = 1 if C = 0, else 0 (C = 1)

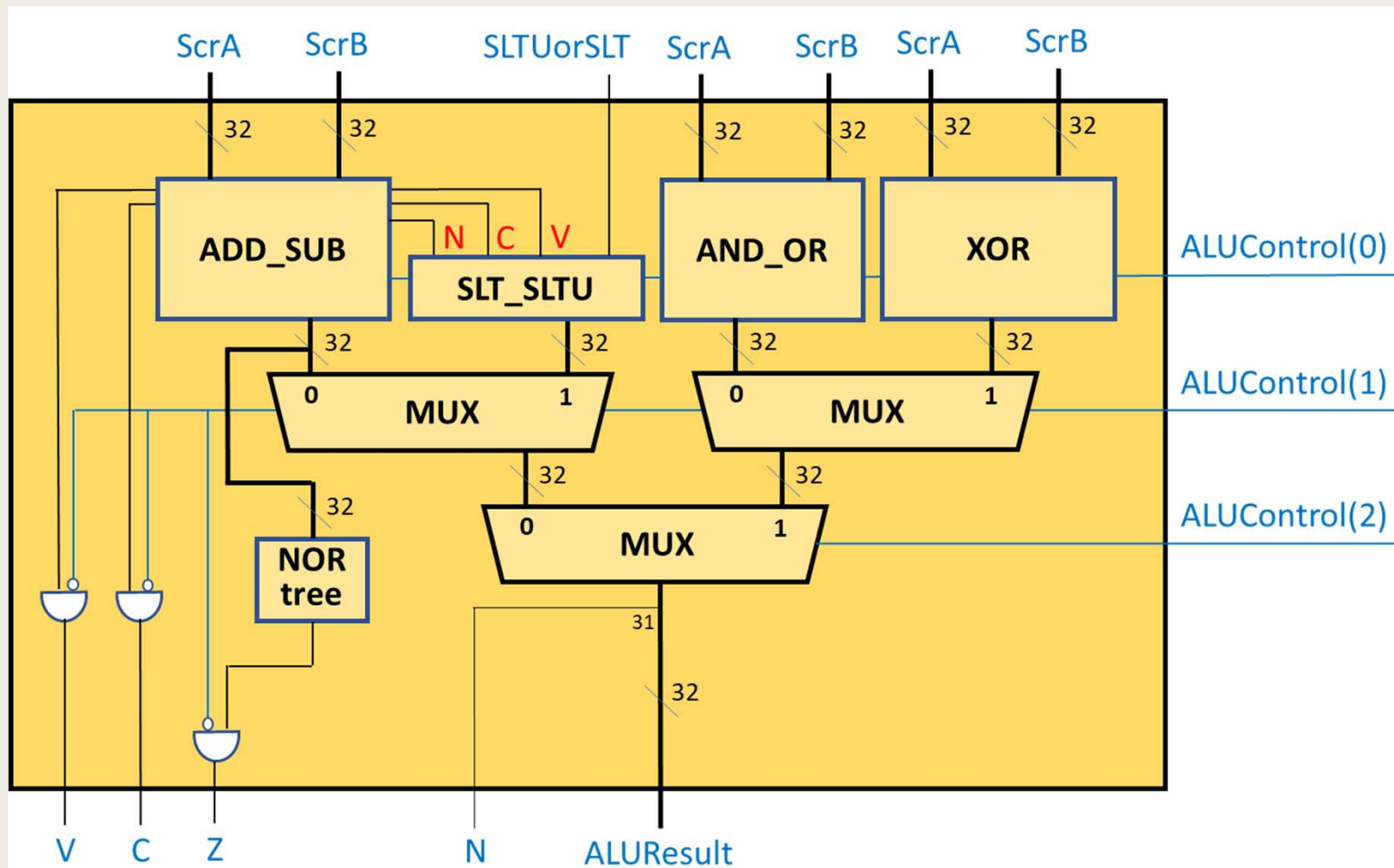
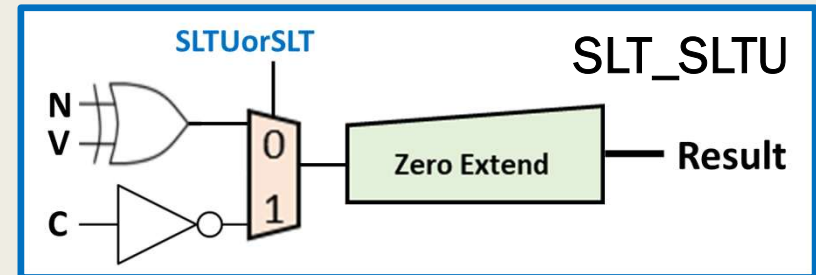
31:25	24:20	19:15	14:12	11:7	6:0
funct7	rs2	rs1	funct3	rd	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

- Απαιτούμενες λειτουργίες των εντολών SLT/SLTU που ήδη υποστηρίζονται:
 - ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs1$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD1$
 - ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs2$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD2$
 - Εκτελείται η πράξη της αφαίρεσης ($ALUControl(0)=1$) και παράγονται τα ALU flags N, C, V
 - Το αποτέλεσμα των πράξεων στη μονάδα ALU ($ALUResult$) γράφεται ετεροχρονισμένα στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
 - ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση της αμέσως επόμενης εντολής: $PC' = PC + 4$
- Νέες λειτουργίες για τις εντολές ALU:
 - η υπομονάδα SLT_SLTU της ALU πρέπει να υποστηρίζει την παραγωγή του
 - $Result = N \text{ xor } V$ για την εντολή SLT ($SLTUorSLT=0, ALUControl(2:0)=011$), και
 - $Result = \text{not } C$ για την εντολή SLTU ($SLTUorSLT=1, ALUControl(2:0)=011$)
 - Στη συνέχεια απαιτείται επέκταση μηδενός από το 1 bit ($Result$) στα 32 bit, ώστε να παραχθεί το $ALUResult(31:0)$

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

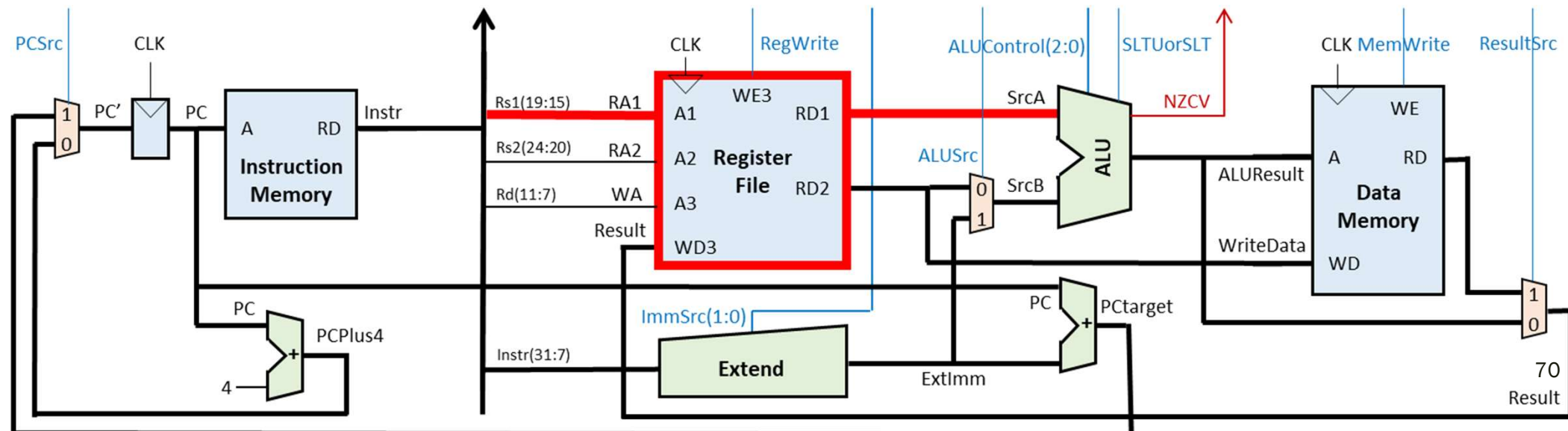
- Υπομονάδα SLT_SLTU της μονάδας ALU



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

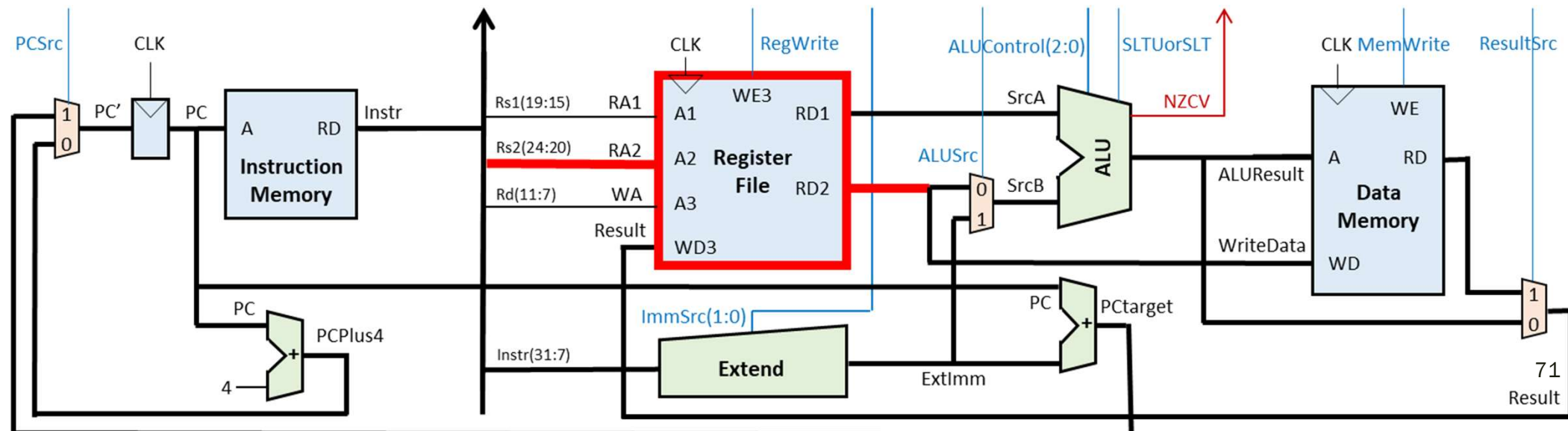
- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης **RD1**
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

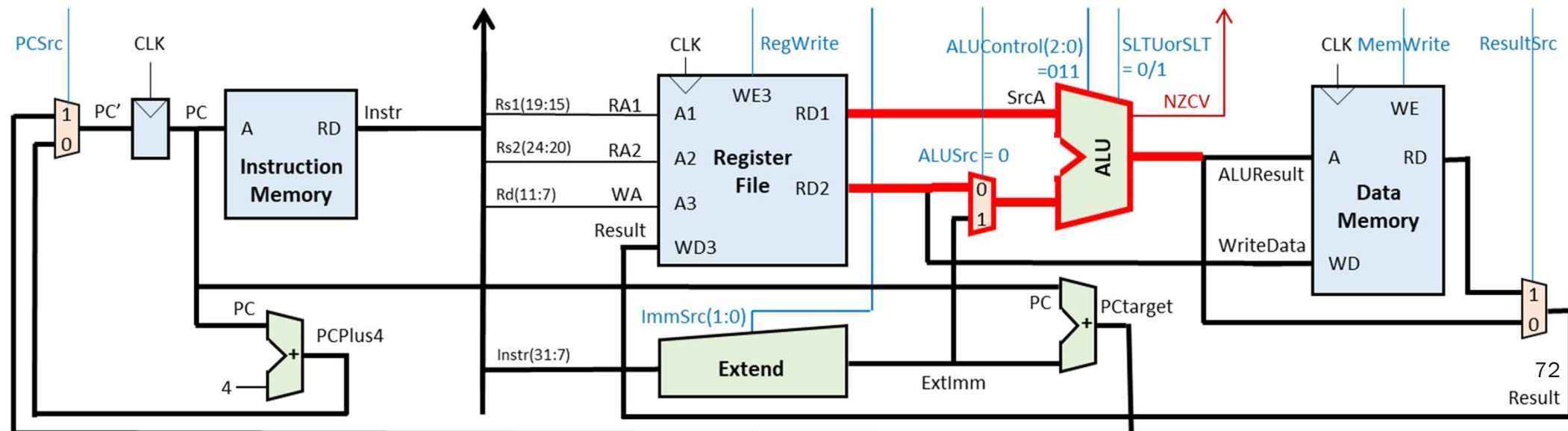
- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs2** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD2
 - σύνδεση του πεδίου *Rs2* της εντολής (*Instr24:20*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD2*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

Εκτέλεση πράξης στη μονάδα ALU με διευθυνσιοδότηση καταχωρητή

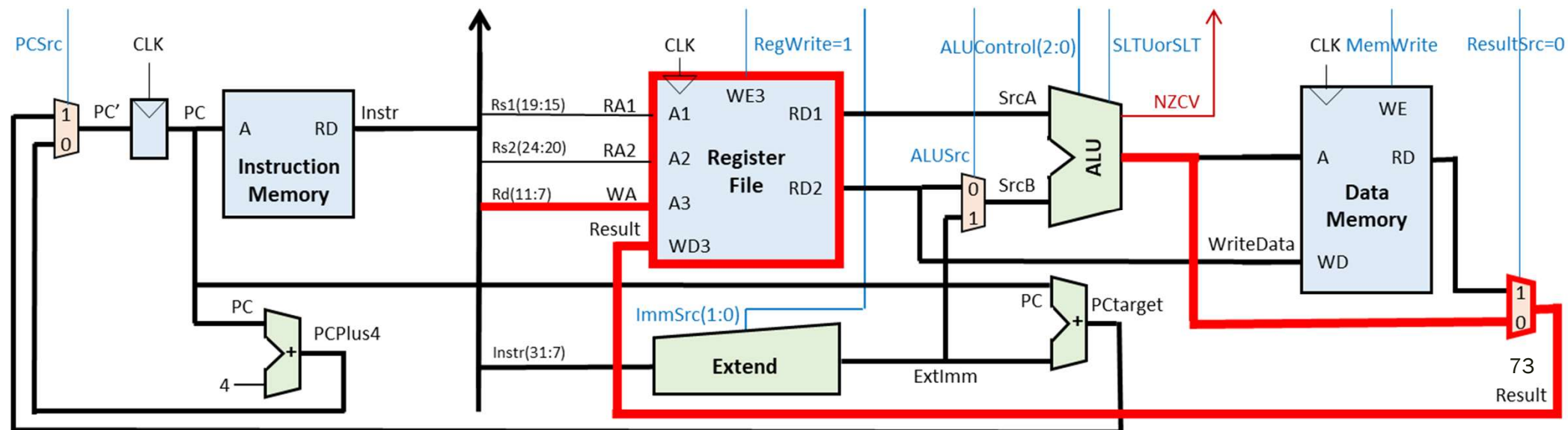
- Η μονάδα ALU εκτελεί αρχικά την πράξη της αφαίρεσης (με **ALUControl(0)=1**) μετά από:
 - σύνδεση της θύρας *RD1* του αρχείου καταχωρητών με την είσοδο *SrcA* της μονάδας ALU
 - σύνδεση της θύρας *RD2* του αρχείου καταχωρητών με την είσοδο *SrcB* της μονάδας ALU
- Τα παραγόμενα μετά την αφαίρεση ALU Flags (NCV) στην έξοδο του αθροιστή/αφαιρέτη χρησιμοποιούνται για να παραχθεί η έξοδος *Result* ως εξής:
 - $\text{Result} = N \text{ xor } V$ για την εντολή SLT (**SLTUorSLT=0**, **ALUControl(2:0)=011**), και
 - $\text{Result} = \text{not } C$ για την εντολή SLTU (**SLTUorSLT=1**, **ALUControl(2:0)=011**)
 - Στη συνέχεια απαιτείται επέκταση μηδενός από το 1 bit (*Result*) στο *ALUResult(31:0)*
- Σήματα ελέγχου: **ALUSrc = 0**, **SLTUorSLT=0/1** και **ALUControl(2:0)=011**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (x0-x31)

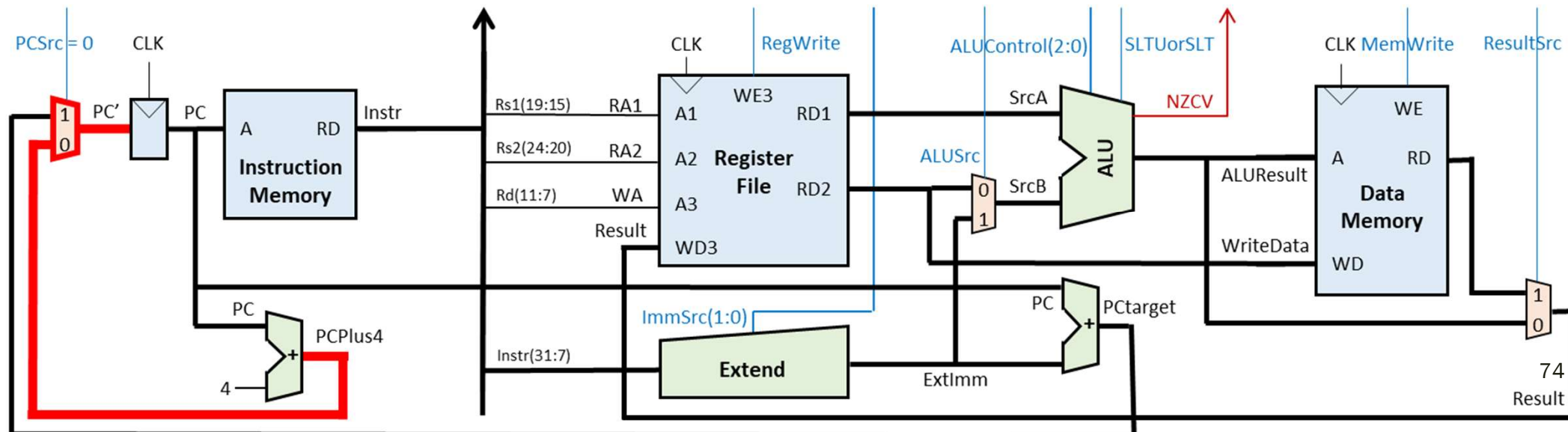
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον **καταχωρητή προορισμού Rd** του αρχείου καταχωρητών
 - σύνδεση του πεδίου *Rd* της εντολής (*Instr11:7*) στην είσοδο διευθύνσεων εγγραφής *A3*
 - σύνδεση της εξόδου *ALUResult* της μονάδας *ALU* με τη θύρα εγγραφής *WD3* του αρχείου καταχωρητών μέσω του πολυπλέκτη *Result*
 - σύγχρονη εγγραφή περιεχομένου καταχωρητή από τη θύρα εγγραφής *WD3*
 - σήματα ελέγχου: **ResultSrc = 0** και **RegWrite = 1** (εγγραφή αρχείου καταχωρητών)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

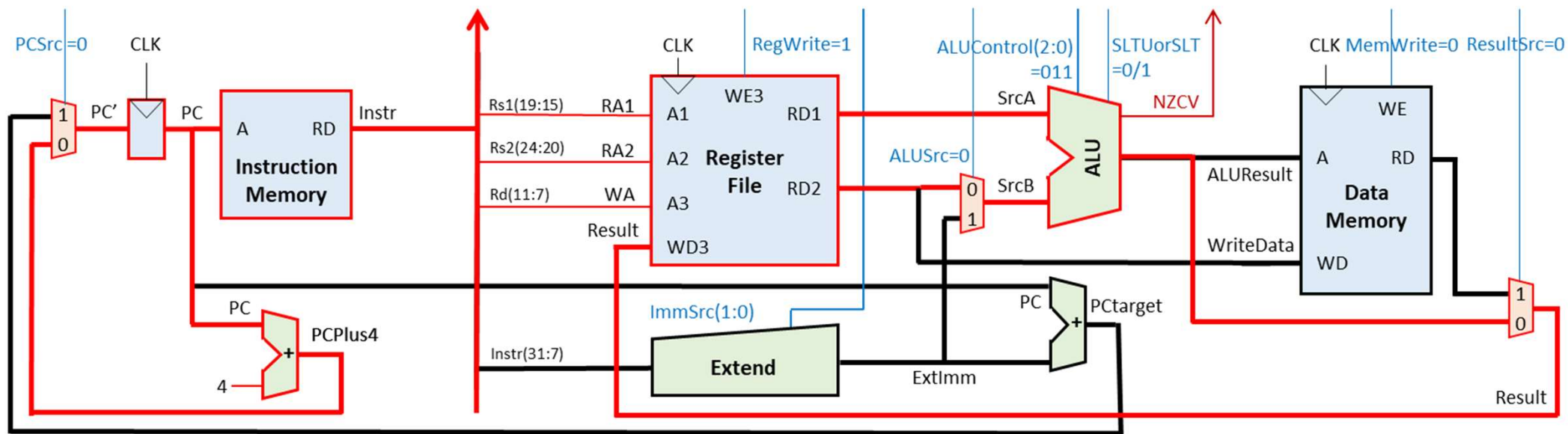
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει την αμέσως επόμενη εντολή, $PC' = PC + 4$
 - σύνδεση της εισόδου 0 του πολυπλέκτη $PCSrc$ με την έξοδο $PCPlus4$ του αθροιστή κατά 4
 - σύνδεση της εξόδου του πολυπλέκτη $PCSrc$ με την είσοδο PC' του μετρητή PC
 - σήματα ελέγχου: $PCSrc = 0$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

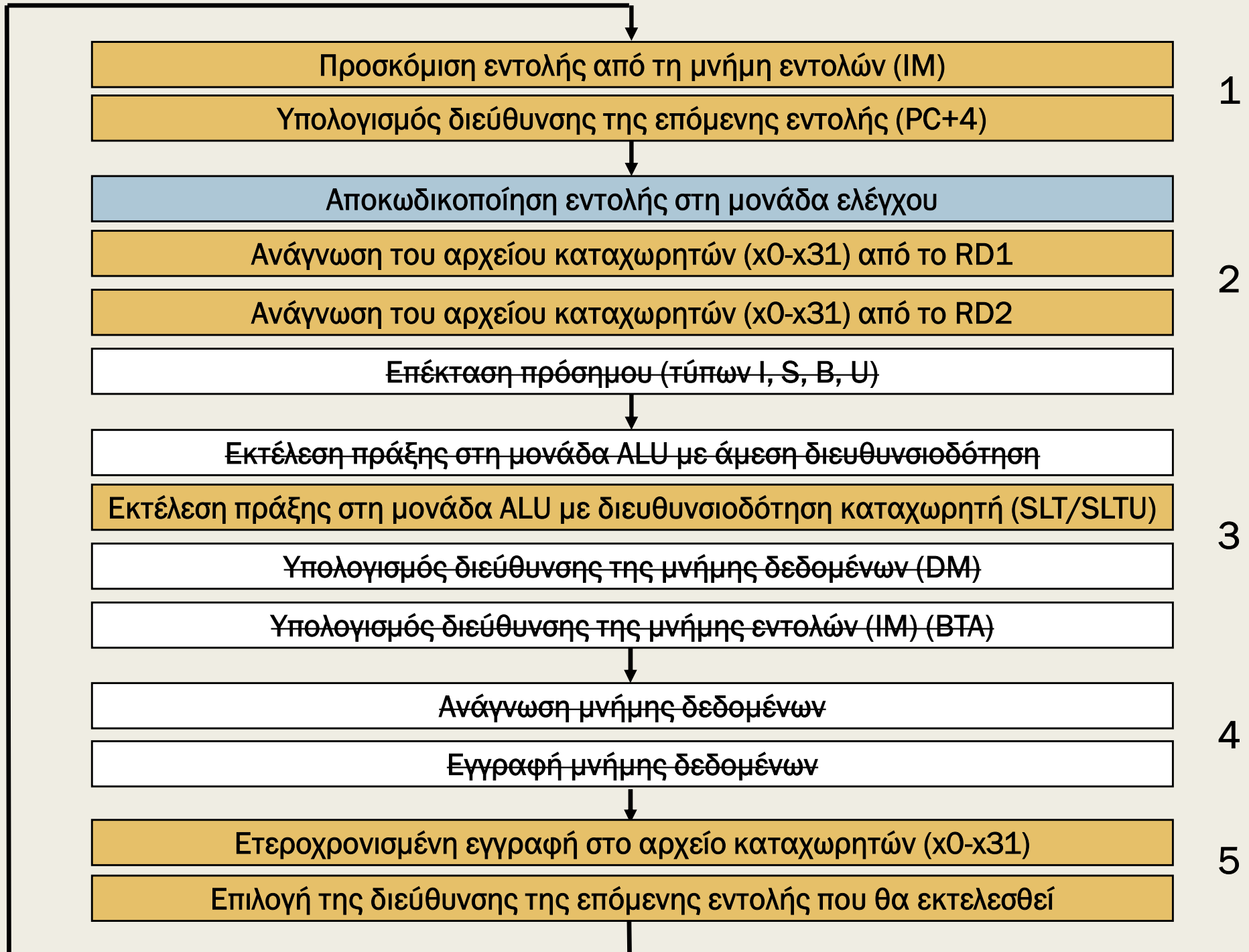


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

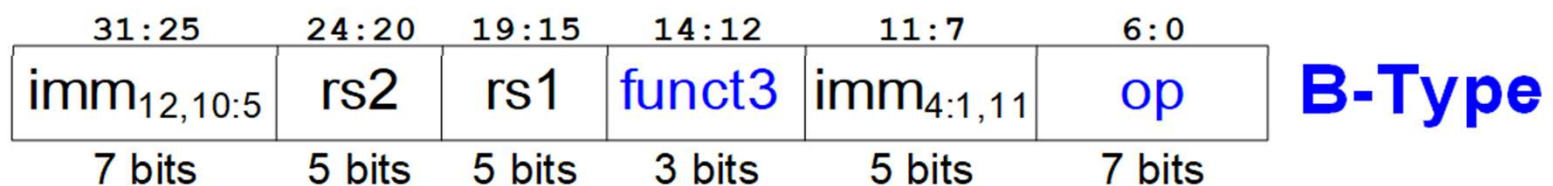


Λειτουργίες εντολών SLT/SLTU (8)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- Στη συνέχεια, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί τις **εντολές διακλάδωσης BEQ και BNE**
 - Κώδικας συμβολικής γλώσσας των εντολών BEQ/BNE:
 - **BEQ Rs1, Rs2, label branch BTA if Rs1 = Rs2**
 - **BNE Rs1, Rs2, label branch BTA if Rs1 ≠ Rs2**
 - Στη μορφή των εντολών ορίζονται:
 - οι διευθύνσεις του **πρώτου καταχωρητή προέλευσης Rs1** και του **δεύτερου καταχωρητή προέλευσης Rs2**
 - ένας **προσημασμένος άμεσος τελεστέος imm(12:1)** των **12 bit** για τον υπολογισμό του **Branch Target Address (BTA)** ($instr(31) \& instr(7) \& instr(30:25) \& instr(11:8)$)
- $$BTA = PC + \text{Sign_Ext}(instr(31) \& instr(7) \& instr(30:25) \& instr(11:8) \& '0')$$
- Απαιτείται **επέκταση πρόσημου τύπου B** στη μονάδα Extend και η εκτέλεση της πράξης της **πρόσθεσης** σε διακριτό αθροιστή εκτός της μονάδας ALU
 - Στη μονάδα ALU εκτελείται η σύγκριση ως αφαίρεση



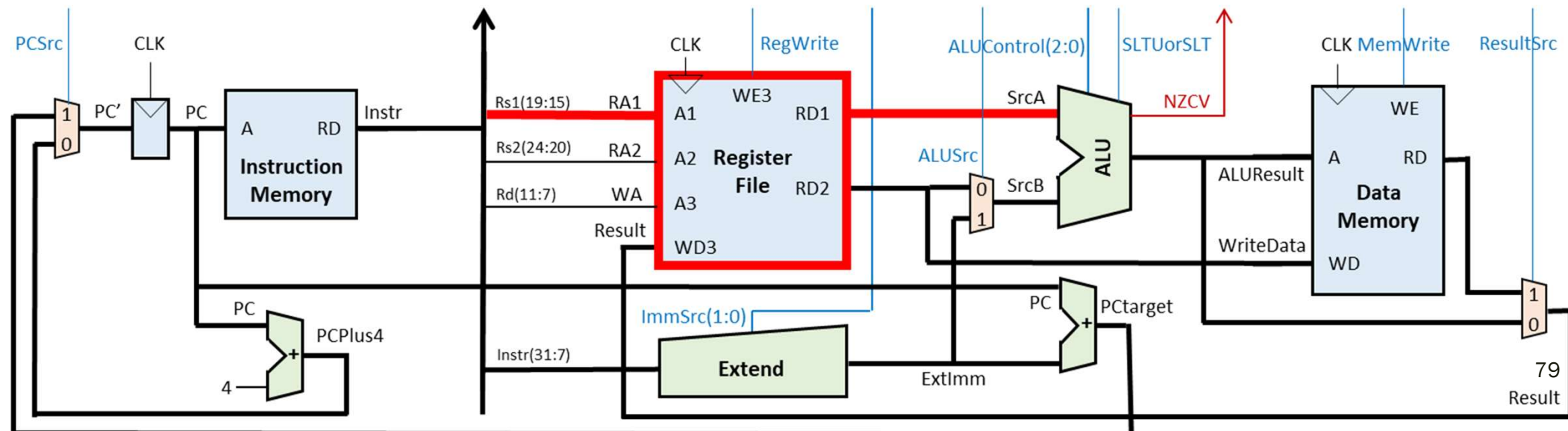
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- Απαιτούμενες λειτουργίες των εντολών Branch που ήδη υποστηρίζονται:
 - ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs1$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD1$
 - ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης $Rs2$ του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης $RD2$
 - η μονάδα ALU εκτελεί την πράξη της αφαίρεσης ($Rs1-Rs2$) και ενημερώνει τα ALU flags (NZCV)
 - Για τις εντολές BEQ ($Z=1$) και BNE ($Z=0$) απαιτείται μόνο το Z (zero)
- Νέες λειτουργίες για τις εντολές Branch:
 - επέκταση προσήμου τύπου B του άμεσου τελεστή $imm(12:0)$ από τα 13 bit στα 32 bit για τον υπολογισμό του BTA
 - $Imm(12:0) = instr(31) \& instr(7) \& instr(30:25) \& instr(11:8) \& '0'$
 - Υπολογισμός του BTA ως $BTA = PC + sign_exted(imm(12:0))$ σε διακριτό αθροιστή
 - ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση BTA όταν ικανοποιείται η συνθήκη Branch ($PC' = BTA$), αλλιώς την διεύθυνση της αμέσως επόμενης εντολής ($PC' = PC + 4$)
 - το αν ικανοποιείται ή όχι η συνθήκη του Branch εξετάζεται σε συγκεκριμένη υπομονάδα της μονάδας ελέγχου που παράγει το σήμα ελέγχου **PCSrc** με βάση τα ALU flags

Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

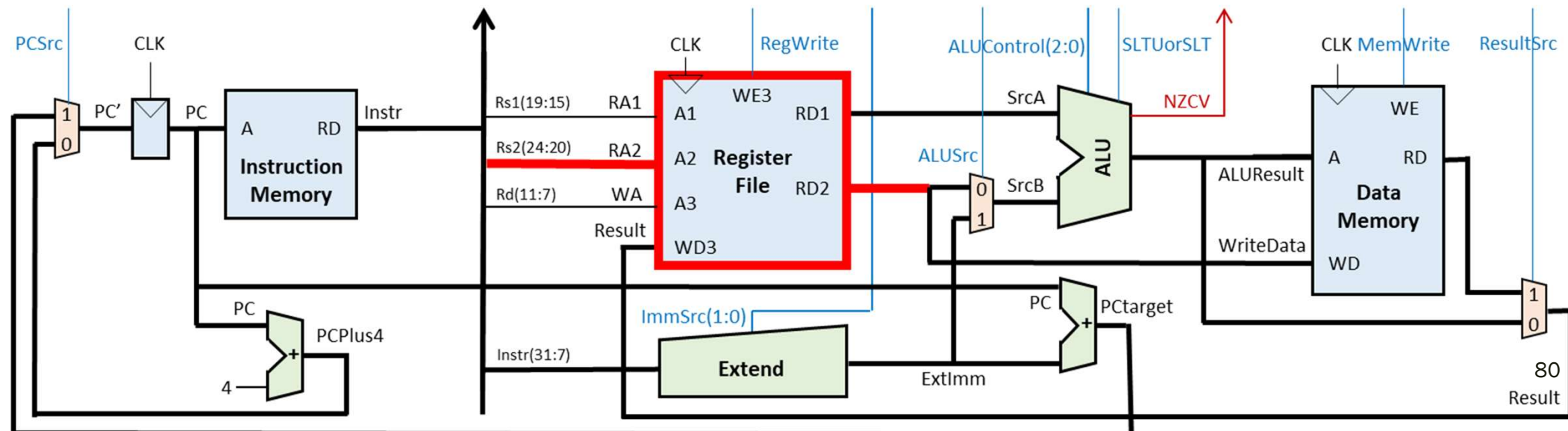
- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs1** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης **RD1**
 - σύνδεση του πεδίου *Rs1* της εντολής (*Instr19:15*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD1*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

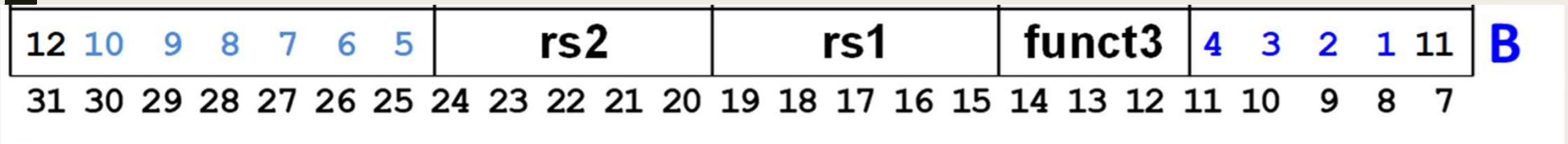
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

- ο τελεστής που είναι αποθηκευμένος στον καταχωρητή προέλευσης **Rs2** του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD2
 - σύνδεση του πεδίου *Rs2* της εντολής (*Instr24:20*) στην είσοδο διευθύνσεων ανάγνωσης *A1*
 - ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης *RD2*
 - σήματα ελέγχου: **RegWrite = 0/1**
(γίνεται ανάγνωση του αρχείου καταχωρητών ανεξάρτητα από την τιμή του **RegWrite**, η τιμή του είναι 1 στην περίπτωση που υπάρχει εγγραφή στον ίδιο κύκλο)

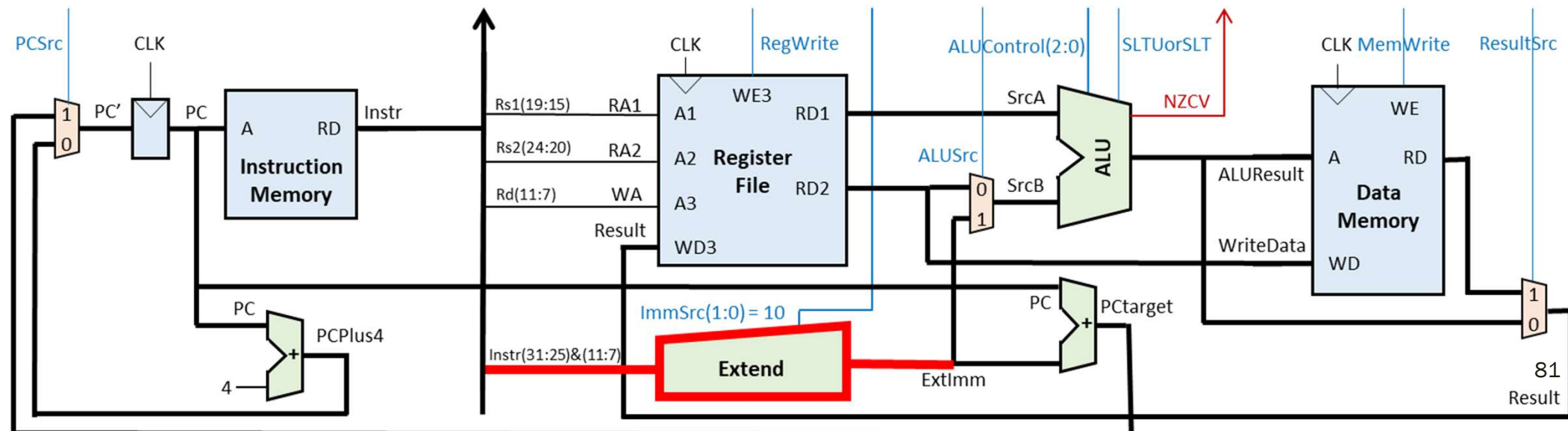


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

Επέκταση πρόσημου (τύπων I, S, B, U) – Επιλογή τύπου B



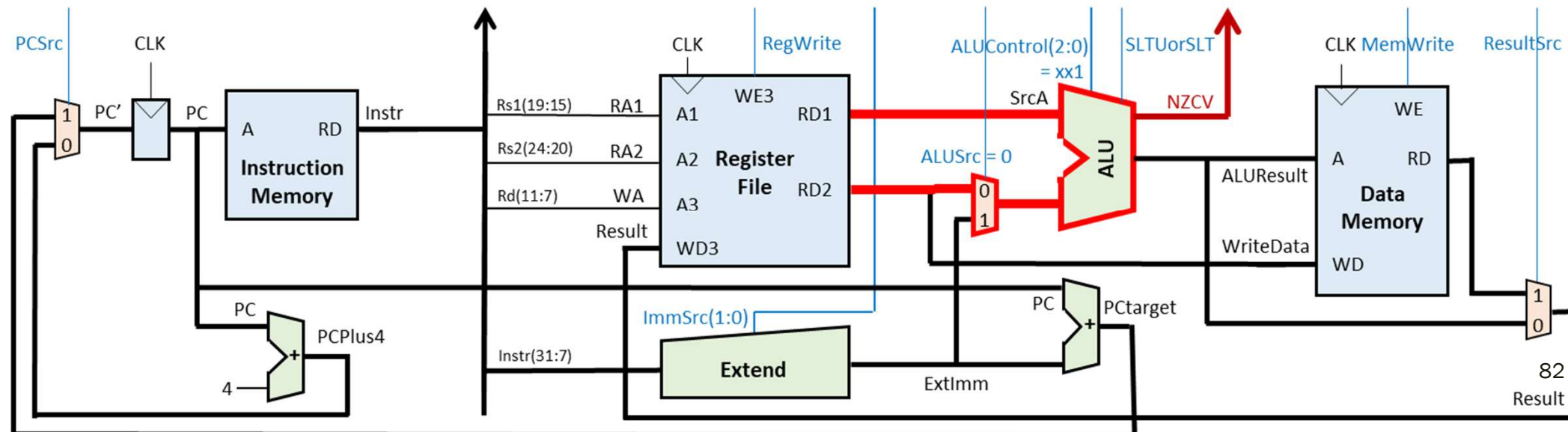
- Η εντολές Branch απαιτούν το υπολογισμό της Branch Target Address (BTA) που προκύπτει με επέκταση προσήμου από τα 13 bit στα 32 bit στην μονάδα Extend
 - $\text{Extend}(31:0) = \text{Sign_Ext}(\text{instr}(31) \& \text{instr}(7) \& \text{instr}(30:25) \& \text{instr}(11:8) \& '0')$
 - Για επέκταση προσήμου τύπου B τα σήματα ελέγχου είναι: **ImmSrc(1:0) = 10**



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

Εκτέλεση πράξης στη μονάδα ALU με διευθυνσιοδότηση καταχωρητή

- Η μονάδα ALU εκτελεί αρχικά την πράξη της αφαίρεσης (με **ALUControl(0)=1**) μετά από:
 - σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA της μονάδας ALU
 - σύνδεση της θύρας RD2 του αρχείου καταχωρητών με την είσοδο SrcB της μονάδας ALU
- Τα παραγόμενα μετά την αφαίρεση ALU Flags (NZCV) στην έξοδο της μονάδας ALU οδηγούνται στη μονάδα ελέγχου η οποία αποφασίζει αν ικανοποιείται ή όχι η συνθήκη της εντολής Branch και παράγει το κατάλληλο σήμα ελέγχου PCSrc
- Σήματα ελέγχου: **ALUSrc = 0**, **ALUControl(2:0)=xx1 (-)**

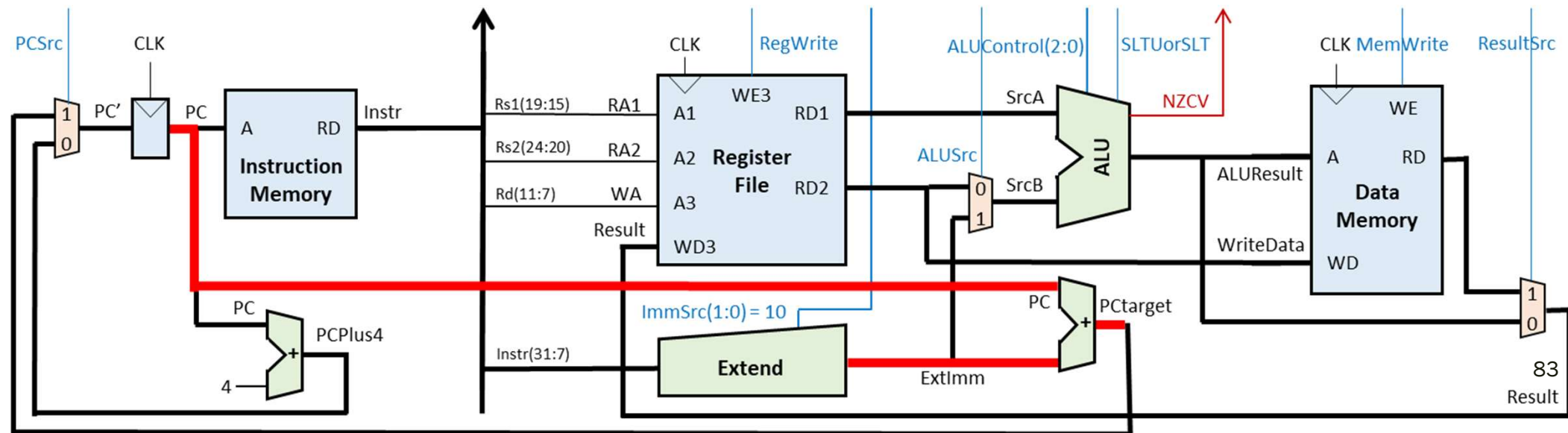


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

Υπολογισμός διεύθυνσης της μνήμης εντολών (IM) (BTA)

- Σε ένα διακριτό αθροιστή υπολογίζεται η διεύθυνση BTA της μνήμης εντολών (IM)
 - σύνδεση της εξόδου PC του μετρητή προγράμματος με την πρώτη είσοδο του διακριτού αθροιστή
 - σύνδεση της εξόδου ExtImm της μονάδας Extend με τη δεύτερη είσοδο του διακριτού αθροιστή
 - εκτελείται στον διακριτό αθροιστή η πρόσθεση ($PC + ExtImm$) και το αποτέλεσμα εμφανίζεται στην έξοδο PCtarget
- Σήματα ελέγχου: Δεν υπάρχουν. Υποθέτουμε ότι $ImmSrc(1:0) = 10$

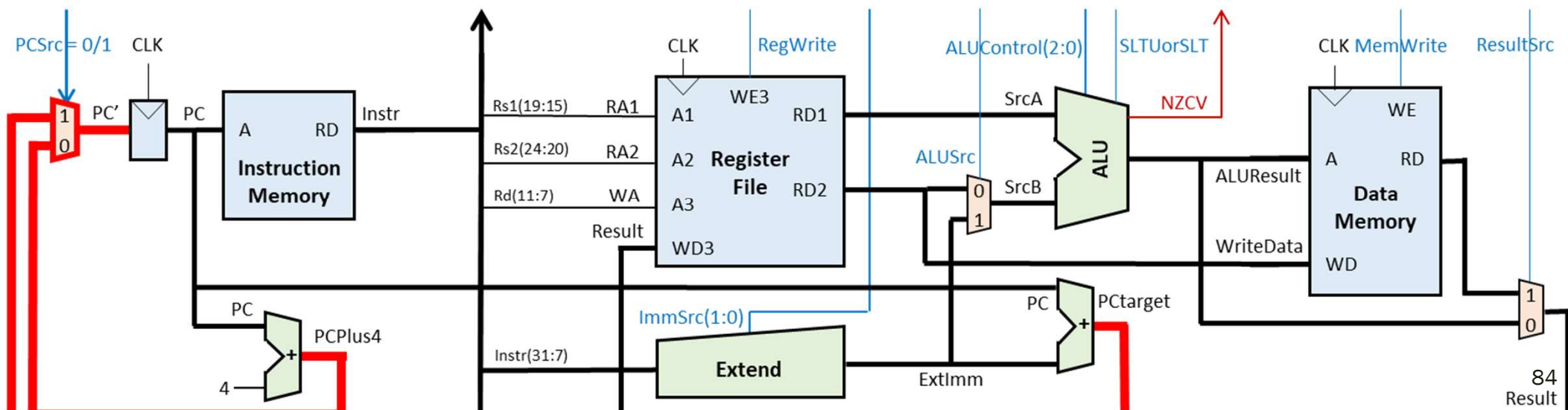
$$BTA = PC + \text{Sign_Ext}(\text{instr}(31) \& \text{instr}(7) \& \text{instr}(30:25) \& \text{instr}(11:8) \& '0')$$



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή Branch

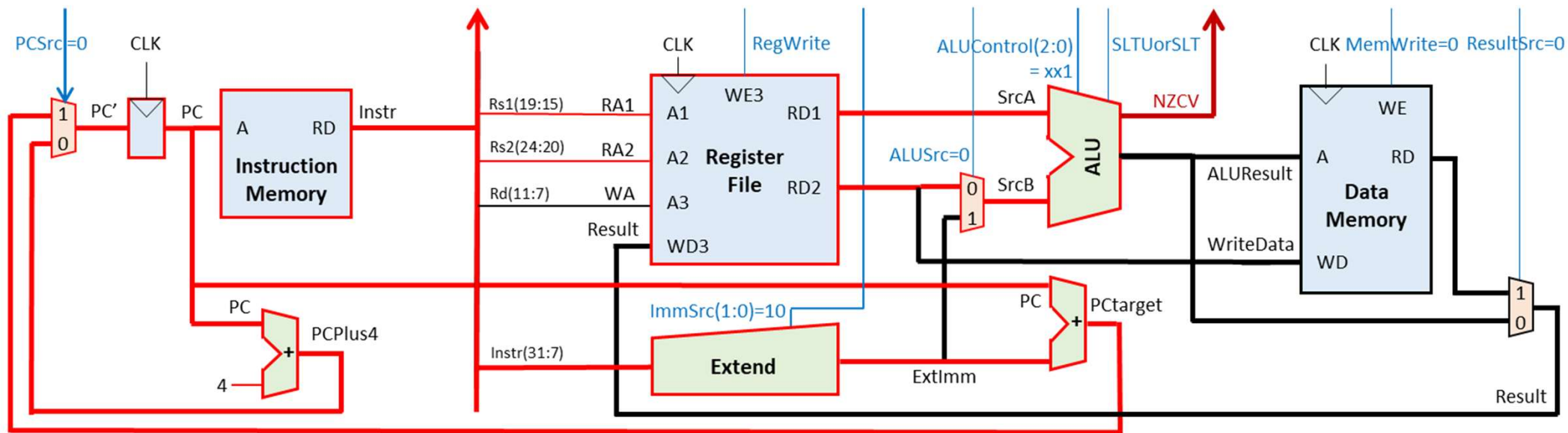
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση:
 - $PC' = PC_{target}$ (BTA), εάν ικανοποιείται η συνθήκη της εντολής Branch ($PCSrc = 1$)
 - $PC' = PC + 4$, εάν δεν ικανοποιείται η συνθήκη της εντολής Branch ($PCSrc = 0$)
- Για τον πολυπλέκτη επιλογής διεύθυνσης επόμενης εντολής υπάρχει:
 - σύνδεση της εισόδου 0 του πολυπλέκτη PCSrc με την έξοδο PCPlus4 του αθροιστή κατά 4
 - σύνδεση της εισόδου 1 του πολυπλέκτη PCSrc με την έξοδο PCtarget του αθροιστή που υπολογίζει την BTA
 - σύνδεση της εξόδου του πολυπλέκτη PCSrc με την είσοδο PC' του μετρητή PC
 - σήμα επιλογής: $PCSrc = 0/1$
- Η νέα διεύθυνση αποθηκεύεται στον μετρητή προγράμματος στην επόμενη ανερχόμενη ακμή του CLK

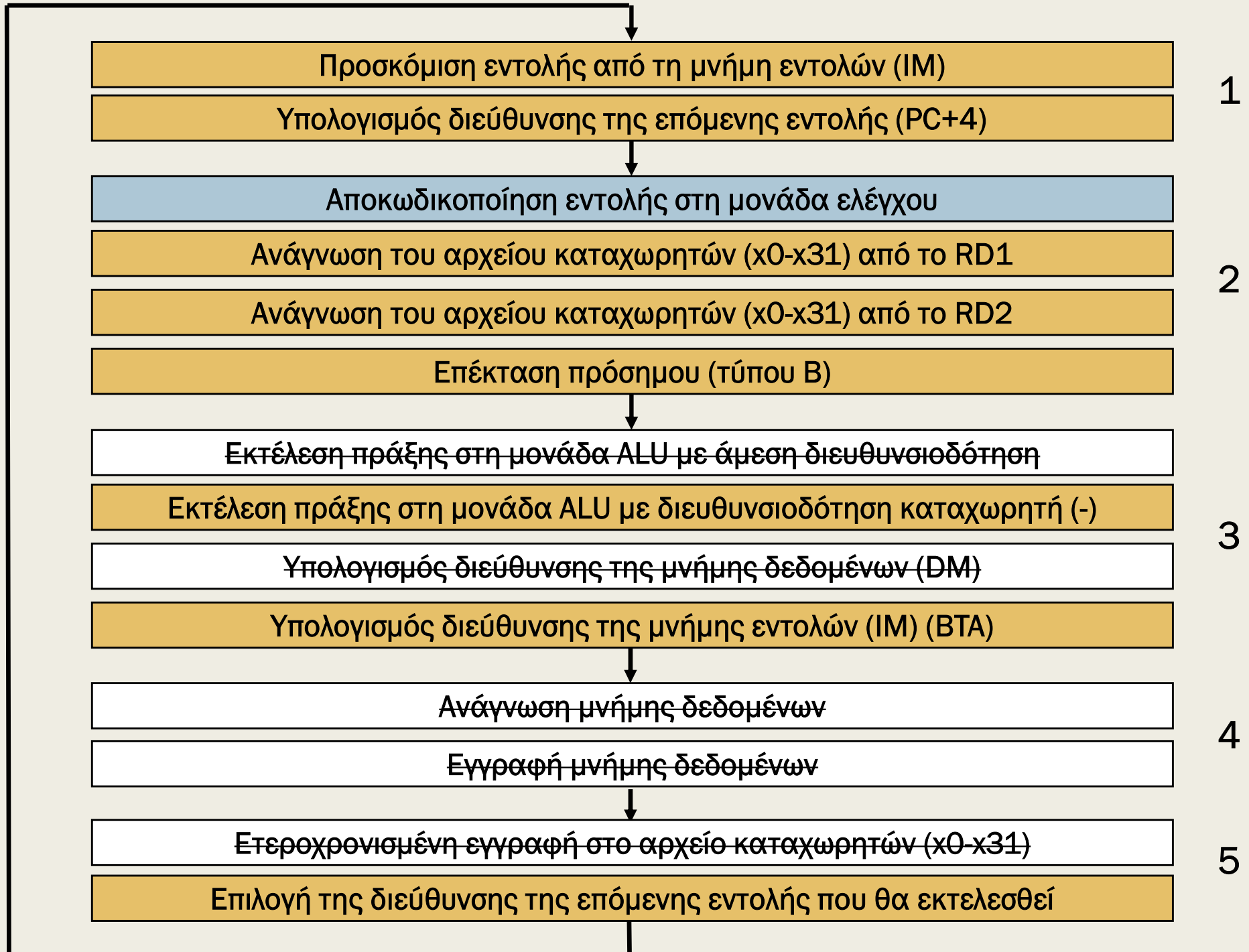


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



Λειτουργίες εντολών Branch (9)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Μονάδα Extend

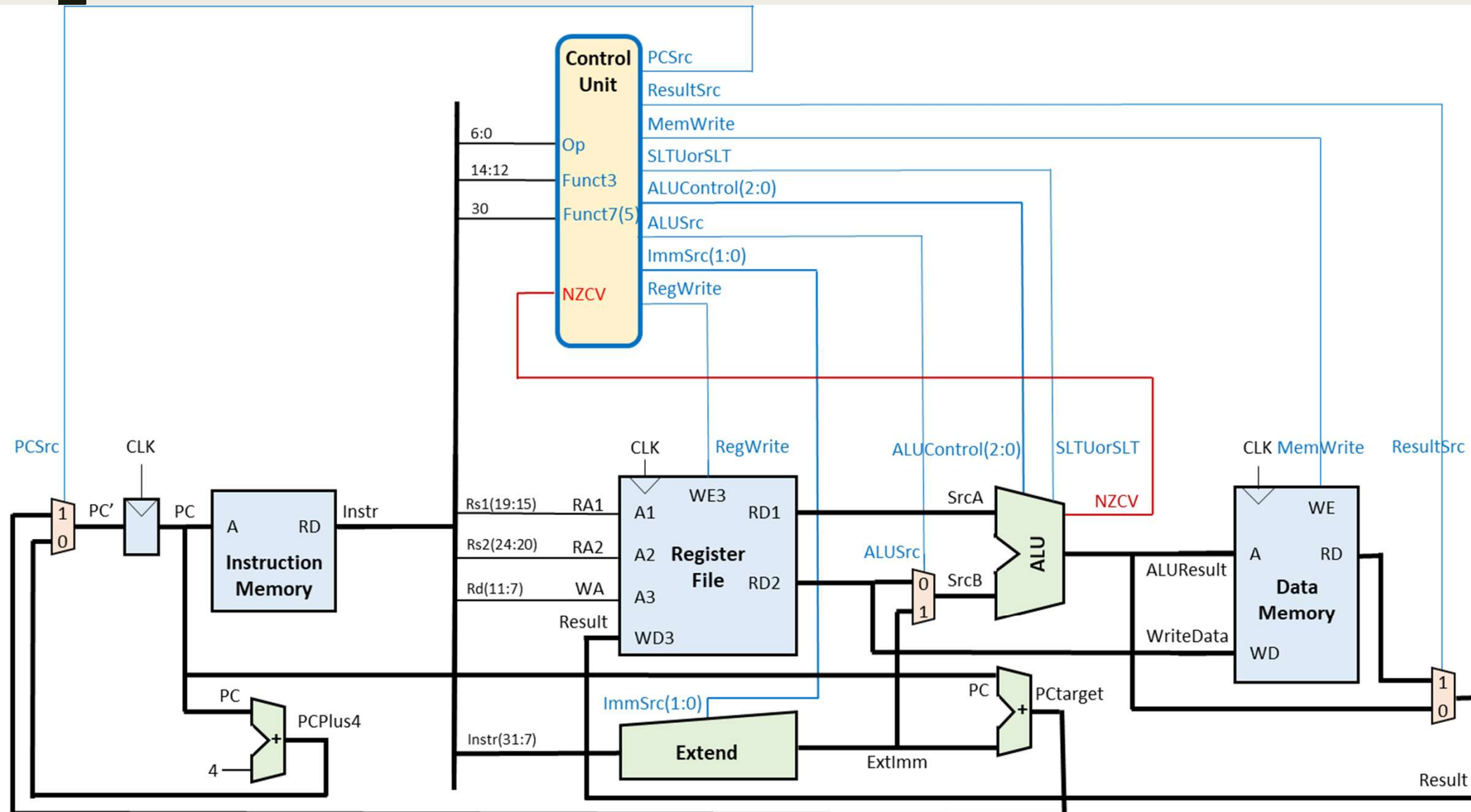
- Για το βασικό σύνολο εντολών της αρχιτεκτονικής RISC-V, που έχουμε μελετήσει μέχρι τώρα, η μονάδα Extend εκτελεί τις ακόλουθες διακριτές επεκτάσεις προσήμου στα 32 bit:

Τύπος	ImmSrc	Extend (31:0) =	Input
I	00	<code>Sign_Ext(instr(31:20))</code>	12 bit
S	01	<code>Sign_Ext(instr(31:25) & instr(11:7))</code>	12 bit
B	10	<code>Sign_Ext(instr(31) & instr(7) & instr(30:25) & instr(11:8) & '0')</code>	13 bit

11 10 9 8 7 6 5 4 3 2 1 0	rs1	funct3	rd	I
11 10 9 8 7 6 5	rs2	rs1	funct3 4 3 2 1 0	S
12 10 9 8 7 6 5	rs2	rs1	funct3 4 3 2 1 11	B
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12	rd			U
20 19 18 17 16 15 14 13 12	rd			J
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7				

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

- Ολοκληρωμένος επεξεργαστής ενός κύκλου



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

- Η **μονάδα ελέγχου** (control unit) παράγει τα κατάλληλα σήματα ελέγχου για τον χρονισμό του επεξεργαστή με τη χρήση:
 - συνδυαστικής λογικής στην περίπτωση επεξεργαστή ενός κύκλου
- Είσοδοι της μονάδας ελέγχου:
 - πεδίο **op** της εντολής (*Instr*(6:0) για τον προσδιορισμό του τύπου της εντολής και του είδους της πράξης
 - πεδίο **funct3** της εντολής (*Instr*(14:12) για τη δήλωση της λειτουργίας/πράξης
 - πεδίο **funct7(5)** της εντολής (*Instr*(30) ως επέκταση του πεδίου **funct3**
- Η μονάδα ελέγχου ελέγχει εάν ικανοποιείται η συνθήκη για τις εντολές Branch και παράγει το κατάλληλο σήμα ελέγχου **PCSrc**
 - Για την υλοποίηση των εντολών BEQ/BNQ ως είσοδος της μονάδας ελέγχου χρησιμοποιείται μόνο το ALU flag **Z** (zero)

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

- Έξοδοι της μονάδας ελέγχου:
 - σήματα επιλογής πολυπλεκτών
 - **ALUSrc** (mux ALU SrcB)
 - **ResultSrc** (mux Result) (για ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών)
 - σήματα ελέγχου λογικής
 - **ALUControl(2:0)** (επιλογή πράξης στη μονάδα ALU)
 - **SLTUorSLT** (διάκριση μεταξύ SLT και SLTU στη μονάδα ALU)
 - **ImmSrc(1:0)** (επιλογή επέκτασης προσήμου στη μονάδα Extend)
 - σήματα έγκρισης εγγραφής
 - **RegWrite** (αρχείο καταχωρητών)
 - **MemWrite** (μνήμη δεδομένων)
 - σήμα επιλογής διεύθυνσης επόμενης εντολής
 - **PCSrc** (mux PC') – ελέγχεται από το ALU flag **Z** για BEQ/BNE

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

■ Μελέτη των εισόδων της μονάδας ελέγχου: πεδίο **op**

- **op(0:1)** = 11 (πάντα)
- **op(6:2)** = 00000 (στην εντολή LW)
- **op(6:2)** = 00100 (σε όλες τις εντολές ALU τύπου I)
- **op(6:2)** = 01000 (στην εντολή SW)
- **op(6:2)** = 01100 (σε όλες τις εντολές ALU τύπου R)
- **op(6:2)** = 01101 (στην εντολή LUI)
- **op(6:2)** = 11000 (σε όλες τις εντολές Branch)
- **op(6:2)** = 11001 (στην εντολή JALR)

Εντολή	op(6:0)	Εντολή	op(6:0)
ADD	0110011	ADDI	0010011
SUB	0110011	ANDI	0010011
AND	0110011	ORI	0010011
OR	0110011	XORI	0010011
XOR	0110011	SLLI	0010011
SLL	0110011	SRLI	0010011
SRL	0110011	SRAI	0010011
SRA	0110011	LW	0000011
SLT	0110011	SW	0100011
SLTU	0110011	BEQ	1100011
		BNE	1100011
		JALR	1100111
		LUI	0110111

Προκύπτουν 7 διαφορετικές περιπτώσεις εντολών

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

- Μετασχηματισμός του πεδίου $op(6:2)$ των 5 bit στο εσωτερικό σήμα **rop(2:0)** (reduced op) των 3 bit

- $op(0:1) = 11$ (πάντα)
- $op(6:2) = 00000$ (στην εντολή LW)
- $op(6:2) = 00100$ (σε όλες τις εντολές ALU τύπου I)
- $op(6:2) = 01000$ (στην εντολή SW)
- $op(6:2) = 01100$ (σε όλες τις εντολές ALU τύπου R)
- $op(6:2) = 01101$ (στην εντολή LUI)
- $op(6:2) = 11000$ (σε όλες τις εντολές Branch)
- $op(6:2) = 11001$ (στην εντολή JALR)

Εντολή	$op(6:2)$	$rop(2:0)$
LW	00000	000
ALUI	00100	001
SW	01000	010
ALU	01100	011
Branch	11000	100
JALR	11001	101
LUI	01101	11X

Οι εντολές που έχουν κοινό **rop(2,0)** ξεχωρίζουν με βάση τα πεδία **funct3** και **funct7(5)**

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

- Στη συνέχεια θα σχεδιάσουμε τη μονάδα ελέγχου που απαρτίζεται από τις ακόλουθες υπομονάδες:

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής (*Instr*(6:2) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc**(1:0), τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών **RegWrite** και στη μνήμη δεδομένων **MemWrite** και το εσωτερικό σήμα **rop**(2:0)

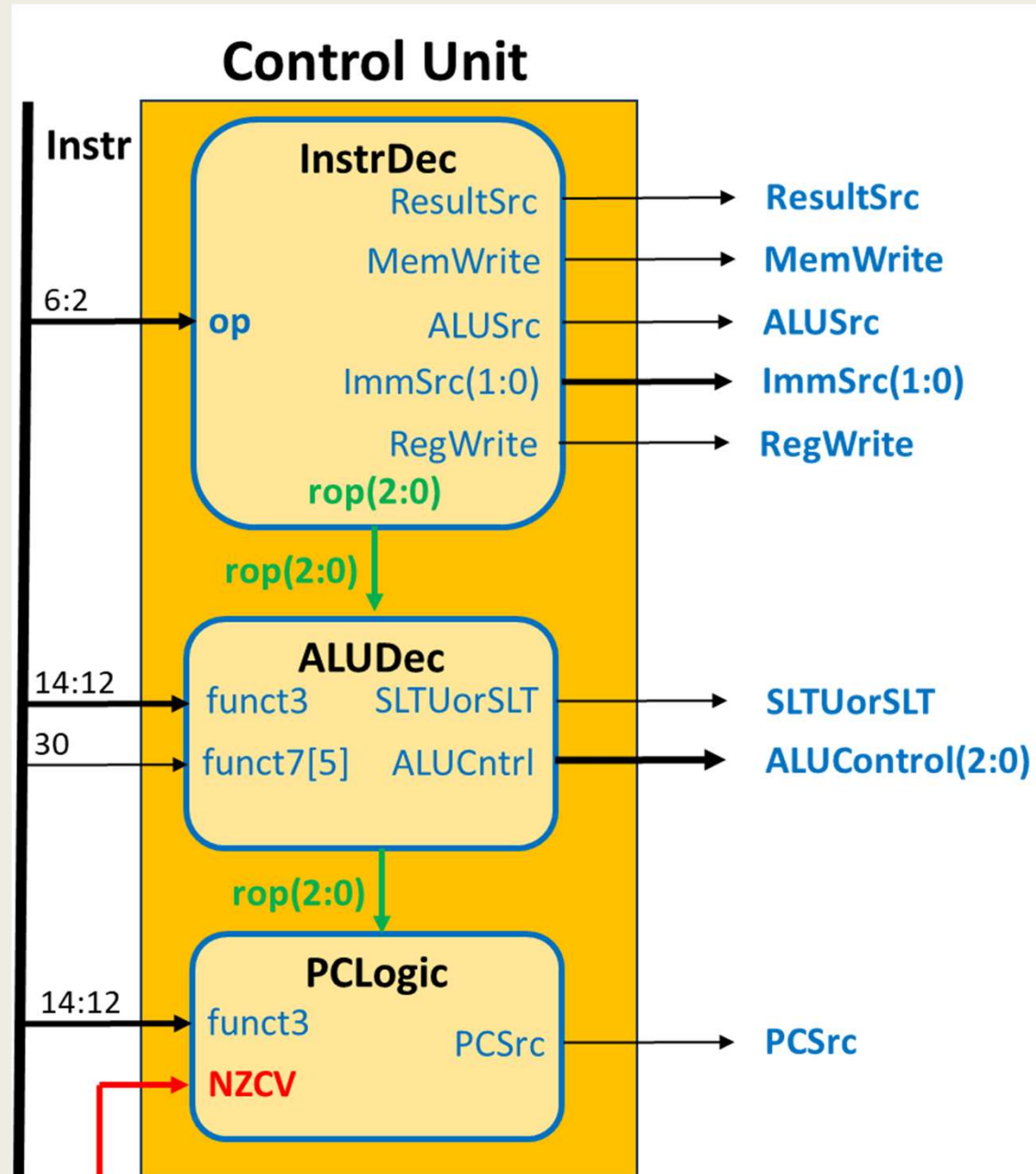
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop**(2:0), καθώς και τα πεδία **funct3** (*Instr*(14:12) και **funct7**(5) (*Instr*(30) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl**(2:0) και **SLTUorSLT**

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop**(2:0), καθώς και το πεδίο **funct3** (*Instr*(14:12), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές *Branch* με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των *BEQ/BNE*), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc**

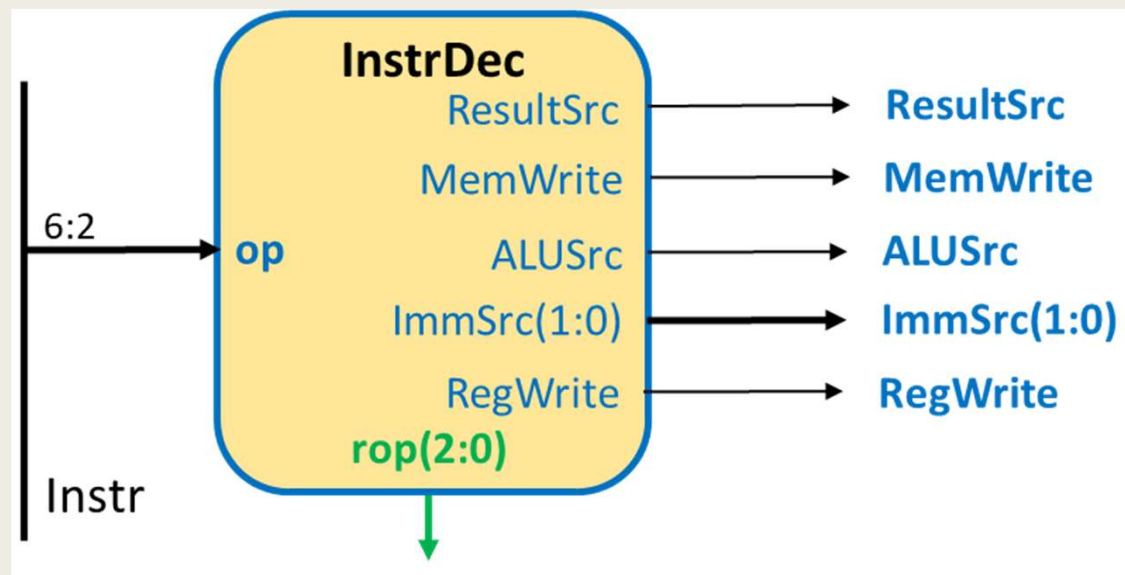
Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής ($\text{Instr}(6:2)$) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc(1:0)**, τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών **RegWrite** και στη μνήμη δεδομένων **MemWrite** και το εσωτερικό σήμα **rop(2:0)**



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

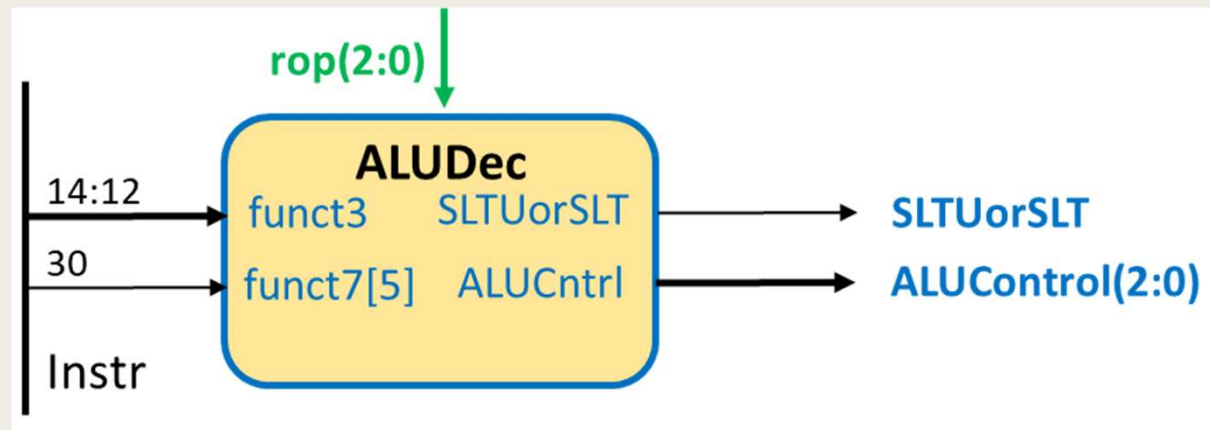
Εντολή	op(6:2)	rop(2:0)	Result Src	Mem Write	ALU Src	Imm Src(1:0)	Reg Write
LW	00000	000	1	0	1	00	1
ALUI	00100	001	0	0	1	00	1
SW	01000	010	X	1	1	01	0
ALU	01100	011	0	0	0	XX	1
Branch	11000	100	X	0	0	10	0
JALR	11001	101	TBD	TBD	TBD	TBD	TBD
LUI	01101	11X	TBD	TBD	TBD	TBD	TBD

Τα σήματα **MemWrite** και **RegWrite** έχουν την τιμή 0 όταν δεν χρησιμοποιούνται η μνήμη δεδομένων και το αρχείο καταχωρητών

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και τα πεδία **funct3** (*Instr*(14:12) και **funct7(5)** (*Instr*(30) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl(2:0)** και **SLTUorSLT**



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:2)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (2:0)	SLTUorSLT
ADD	01100	011	000	0	add	000	X
SUB	01100	011	000	1	sub	001	X
AND	01100	011	111	0	and	100	X
OR	01100	011	110	0	or	101	X
XOR	01100	011	100	0	xor	11X	X
SLL	01100	011	001	0	sll	TBD	X
SRL	01100	011	101	0	srl	TBD	X
SRA	01100	011	101	1	sra	TBD	X
SLT	01100	011	010	0	slt	01X	0
SLTU	01100	011	011	0	sltu	01X	1

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

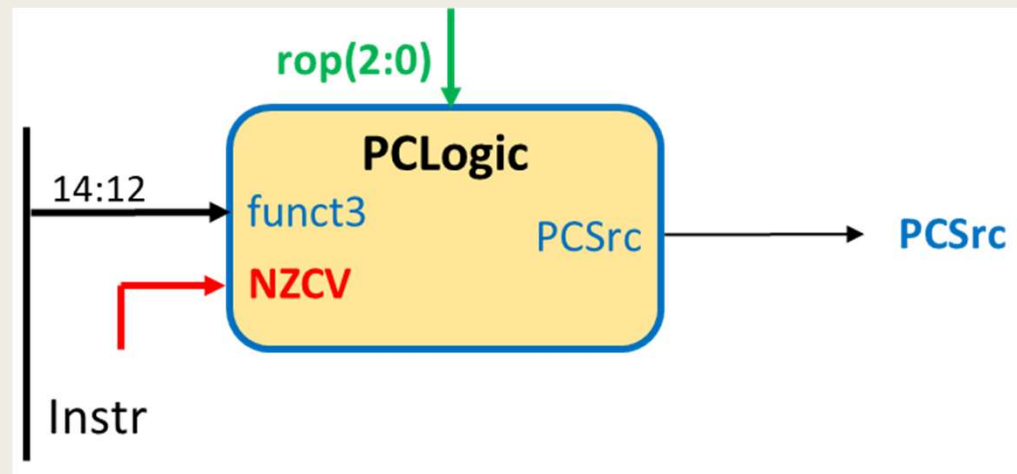
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:2)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (2:0)	SLTUorSLT
ADDI	00100	001	000	X	add	000	X
ANDI	00100	001	111	X	and	100	X
ORI	00100	001	110	X	or	101	X
XORI	00100	001	100	X	xor	11X	X
LW	00000	000	010	X	add	000	X
SW	01000	010	010	X	add	000	X
BEQ	11000	100	000	X	sub	001	X
BNE	11000	100	001	X	sub	001	X
JALR	11001	101	000	X	add	000	X
LUI	01101	11X	XXX	X	-	XXX	X

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και το πεδίο **funct3** (*Instr*(14:12), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές Branch με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των BEQ/BNE), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc**



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

Εντολή	op(6:2)	rop(2:0)	funct3	NZCV	PCSrc
LW	00000	000	XXX	XXXX	0
ALUI	00100	001	XXX	XXXX	0
SW	01000	010	XXX	XXXX	0
ALU	01100	011	XXX	XXXX	0
BEQ	11000	100	000	X0XX	0
BEQ	11000	100	000	X1XX	1
BNE	11000	100	001	X0XX	1
BNE	11000	100	001	X1XX	0
JALR	11001	101	XXX	XXXX	TBD
LUI	01101	11X	XXX	XXXX	TBD

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

■ Εντολές ολίσθησης:

– Τύπου Register (R)

- **SLL** $Rd, Rs1, Rs2$ $Rd = Rs1 \ll Rs2(4:0)$
- **SRL** $Rd, Rs1, Rs2$ $Rd = Rs1 \gg Rs2(4:0)$
- **SRA** $Rd, Rs1, Rs2$ $Rd = Rs1 \ggg Rs2(4:0)$

– Τύπου Immediate (I)

- **SLLI** $Rd, Rs1, uimm$ $Rd = Rs1 \ll imm(4:0)$
- **SRLI** $Rd, Rs1, uimm$ $Rd = Rs1 \gg imm(4:0)$
- **SRAI** $Rd, Rs1, uimm$ $Rd = Rs1 \ggg imm(4:0)$

– Προσθήκες στη μονάδα ALU

- **κυκλώμα ολίσθησης** για αριστερή λογική ολίσθηση (SLL) και δεξιά λογική ολίσθηση (SRL) και δεξιά αριθμητική ολίσθηση (SRA) στη μονάδα ALU (με 3 ολοσθητές)
 - η επιλογή γίνεται με πολυπλέκτη 4 σε 1 με είσοδο επιλογής **ALUControl(1:0)**
- σύνδεση της εισόδου **shamt** του κυκλώματος ολίσθησης, που προσδιορίζει την ποσότητα ολίσθησης, είτε με τα 5 lsb της εισόδου **SrcB** της μονάδας ALU (**SrcB(4:0)**) για εντολές τύπου R, είτε με τα 5 lsb του άμεσου τελεστέου **imm(4:0)** της εντολής (**instr(24:20)**) για εντολές τύπου I.
- επέκταση του **ALUControl(2:0]** της μονάδας ALU από τα 3 bit στα 4 bit, ήτοι **ALUControl(3:0]**

Προσοχή! Εάν θέλουμε να υποστηρίξουμε ταυτόχρονα και τους δύο τύπους ολίσθησης απαιτείται ένας επιπλέον πολυπλέκτης 2 σε 1 των 5 bit πριν την είσοδο **shamt** με επαναχρησιμοποίηση της εισόδου επιλογής **SLTUorSLT (= shiftRorl)**

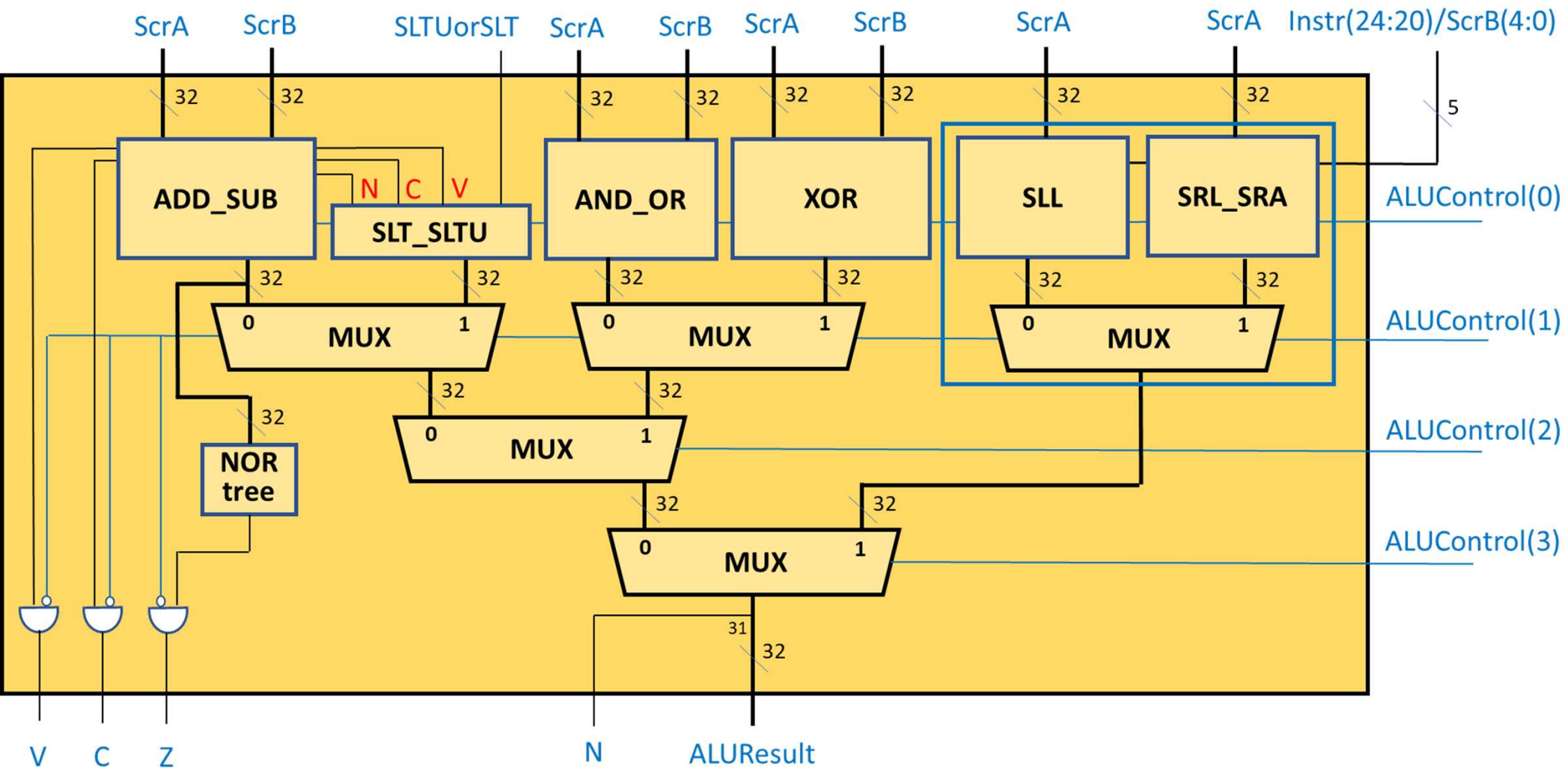
Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

- Πίνακας λειτουργιών της νέας μονάδας ALU

ALUControl(3:0)	Πράξη
0000	Add
0001	Subtract
0011 (SLTUorSLT=0)	SLT
0011 (SLTUorSLT=1)	SLTU
0100	AND
0101	OR
011X	XOR
1X0X	SLL
1X10	SRL
1X11	SRA

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

- Περιγραφή δομής της νέας μονάδας ALU



Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

■ Εντολή LUI:

- Τύπου *Upper immediate (U)*

■ **LUI Rd, upimm Rd = (upimm, 0x000)**

- Προσθήκες στην διαδρομή δεδομένων

- επέκταση της μονάδας Extend, ώστε να υποστηρίζει επέκταση τύπου U

Τύπος	ImmSrc	Extend (31:0) =	Input
I	00	<code>Sign_Ext(instr(31:20))</code>	12 bit
S	01	<code>Sign_Ext(instr(31:25) & instr(11:7))</code>	12 bit
B	10	<code>Sign_Ext(instr(31) & instr(7) & instr(30:25) & instr(11:8) & '0')</code>	13 bit
U	11	<code>instr(31:12) & 0x000</code>	20 bit

- επέκταση του πολυπλέκτη Result σε 3 σε 1, ώστε να επιλέγει μεταξύ:

- 00: ALUResult, 01: RD (από DM), 10 ExtImm για εγγραφή στο αρχείο καταχωρητών (Rd)
- η επιλογή γίνεται με είσοδο επιλογής **ResultSrc(1:0)**

Επεξεργαστής ενός κύκλου: ολοκληρωμένη διαδρομή δεδομένων – Μονάδα Extend

- Για το βασικό σύνολο εντολών της αρχιτεκτονικής RISC-V, που έχουμε μελετήσει μέχρι τώρα, η μονάδα Extend εκτελεί τις ακόλουθες διακριτές επεκτάσεις προσήμου στα 32 bit:

Τύπος	ImmSrc	Extend (31:0) =	Input
I	00	<code>Sign_Ext(instr(31:20))</code>	12 bit
S	01	<code>Sign_Ext(instr(31:25) & instr(11:7))</code>	12 bit
B	10	<code>Sign_Ext(instr(31) & instr(7) & instr(30:25) & instr(11:8) & '0')</code>	13 bit
U	11	<code>instr(31:12) & 0x000</code>	20 bit

11 10 9 8 7 6 5 4 3 2 1 0	rs1	funct3	rd	I
11 10 9 8 7 6 5	rs2	rs1	funct3 4 3 2 1 0	S
12 10 9 8 7 6 5	rs2	rs1	funct3 4 3 2 1 11	B
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12			rd	U
20 10 9 8 7 6 5 4 3 2 1 11 10 10 17 16 15 14 13 12			rd	J
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7				

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

■ Εντολή JALR/JR:

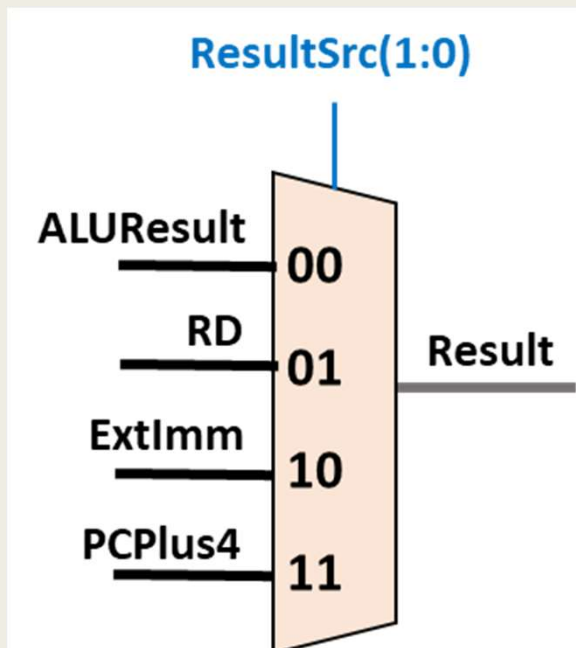
– Τύπου *Immediate (I)*

■ **JALR** **Rd, Rs1, imm** $PC = Rs1 + S_Ext(imm)$
 $Rd = PC + 4$

– Προσθήκες στην διαδρομή δεδομένων

■ επέκταση του πολυπλέκτη Result σε 4 σε 1, ώστε να επιλέγει μεταξύ:

- 00: ALUResult, 01: RD (από DM), 10: ExtImm και 11: PCPlus4 για εγγραφή στο αρχείο καταχωρητών (Rd)
- η επιλογή γίνεται με είσοδο επιλογής **ResultSrc(1:0)**



JR ra -> JALR zero, ra, 0 $PC = ra + 0$
zero = PC + 4

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

■ Εντολή JALR/JR:

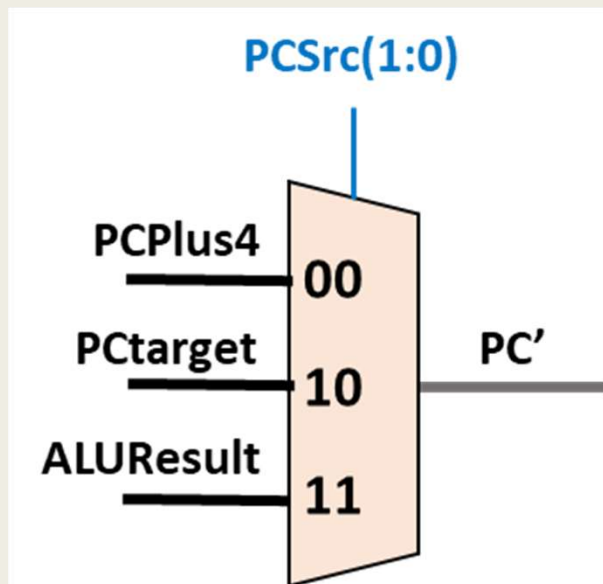
– Τύπου *Immediate (I)*

■ **JALR** **Rd, Rs1, imm** $PC = Rs1 + S_Ext(imm)$
 $Rd = PC + 4$

– Προσθήκες στην διαδρομή δεδομένων

■ επέκταση του πολυπλέκτη PC' σε 4 σε 1, ώστε να επιλέγει μεταξύ:

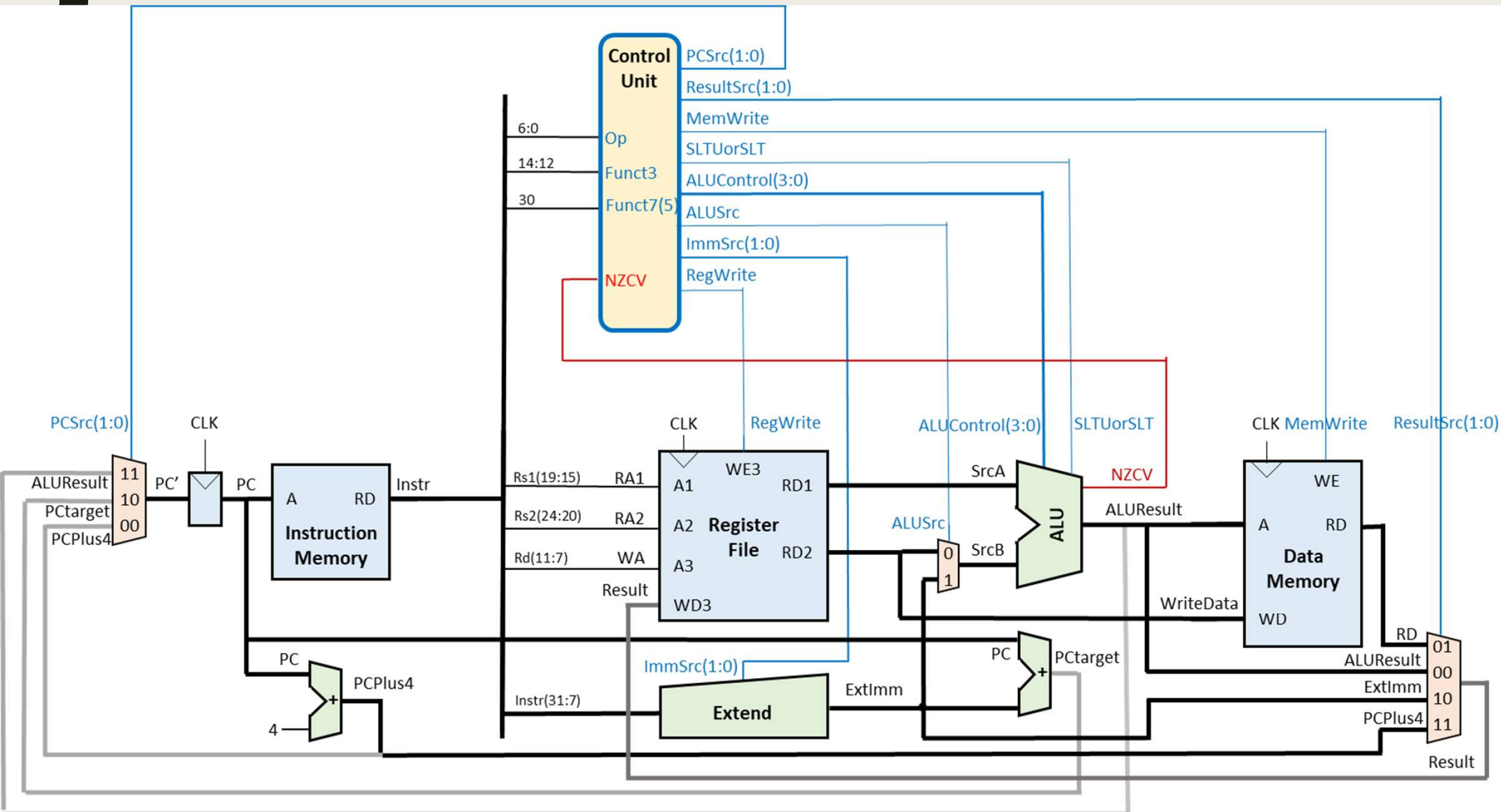
- 00: PCPlus4, 10: PCtarget, 11: ALUResult
για εγγραφή στο μετρητή προγράμματος PC
- η επιλογή γίνεται με είσοδο επιλογής **PCSrc(1:0)**



JR ra -> JALR zero, ra, 0 $PC = ra + 0$
zero = PC + 4

Επεξεργαστής ενός κύκλου: ολοκληρωμένη μικροαρχιτεκτονική RISC-V ενός κύκλου

- Με όλες τις βασικές εντολές (με ολισθήσεις τύπου R)



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

- Στη συνέχεια θα σχεδιάσουμε τη μονάδα ελέγχου που απαρτίζεται από τις ακόλουθες υπομονάδες:

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής ($\text{Instr}(6:2)$) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc(1:0)**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc(1:0)**, τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών **RegWrite** και στη μνήμη δεδομένων **MemWrite** και το εσωτερικό σήμα **rop(2:0)**

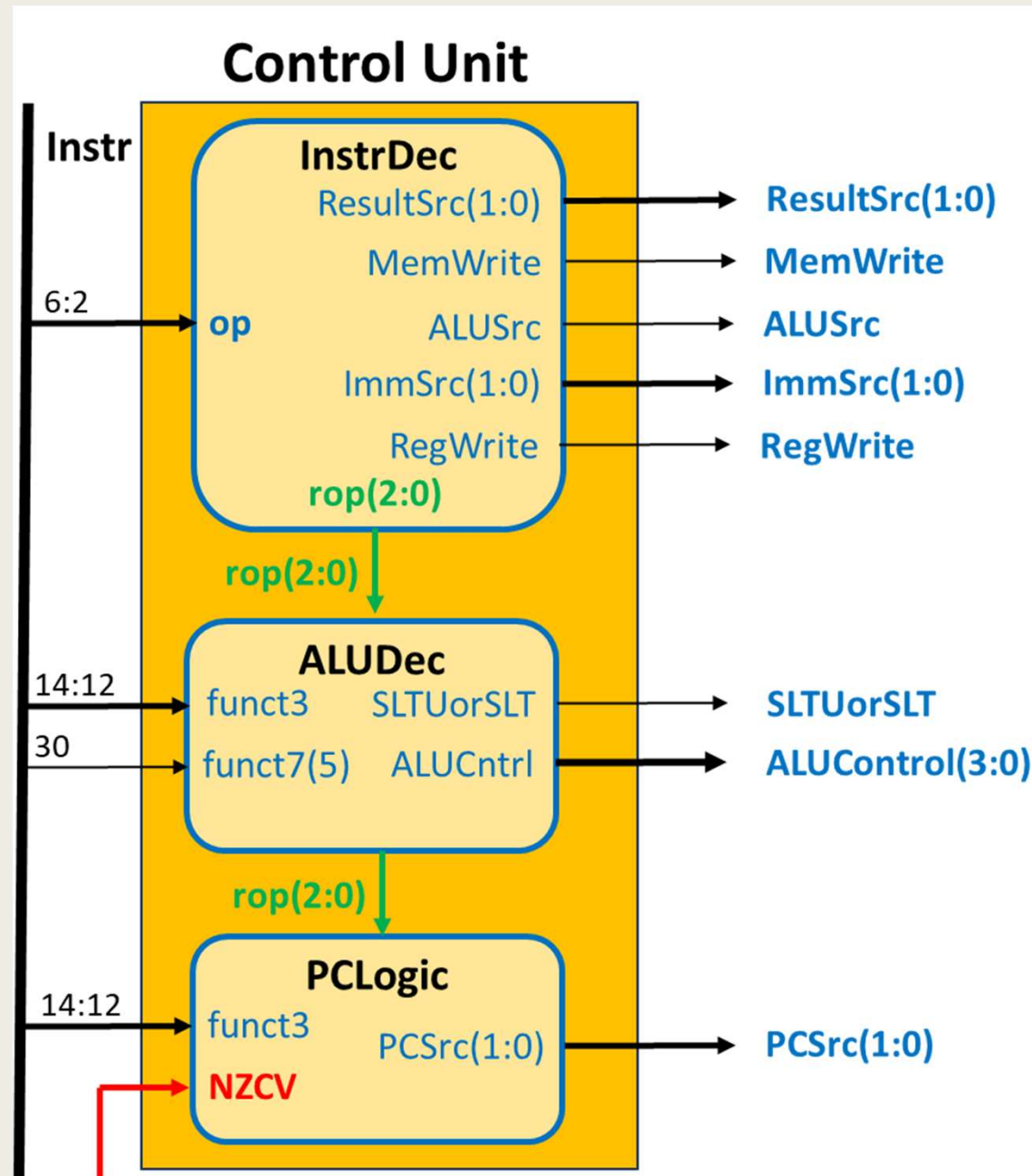
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και τα πεδία **funct3** ($\text{Instr}(14:12)$) και **funct7(5)** ($\text{Instr}(30)$) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl(3:0)** και **SLTUorSLT**

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και το πεδίο **funct3** ($\text{Instr}(14:12)$), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές **Branch** με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των **BEQ/BNE**), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc(1:0)**

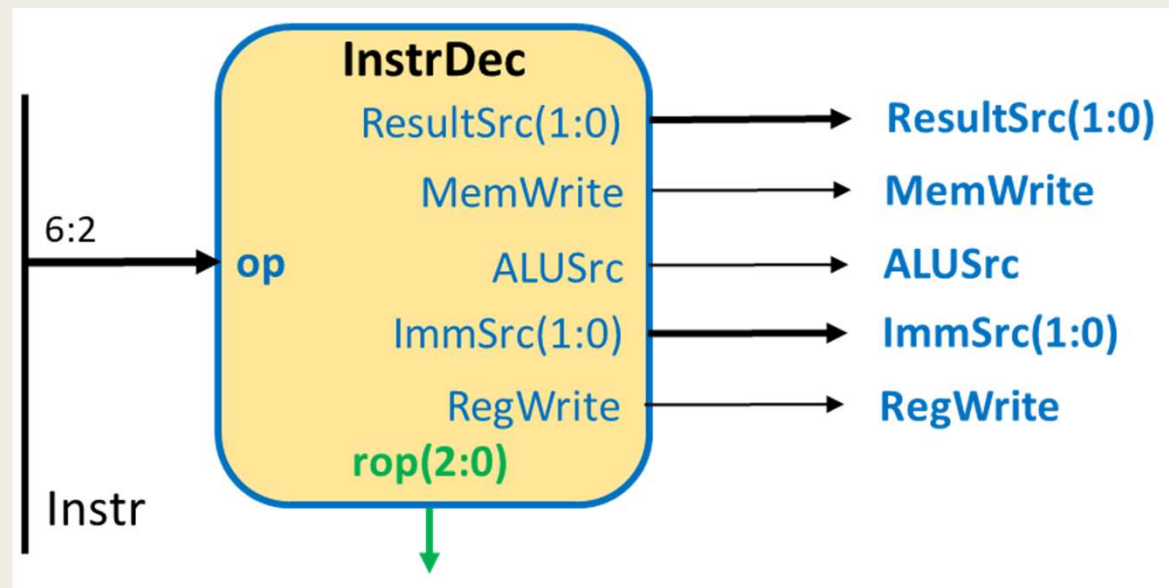
Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής ($\text{Instr}(6:2)$) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc(1:0)**, τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών **RegWrite** και στη μνήμη δεδομένων **MemWrite** και το εσωτερικό σήμα **rop(2:0)**



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

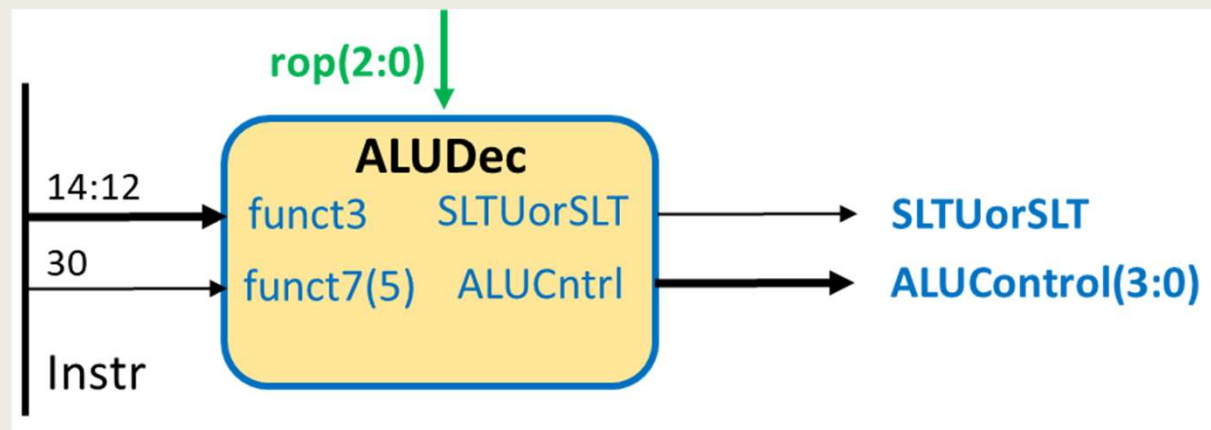
Εντολή	op(6:2)	rop(2:0)	Result Src(1:0)	Mem Write	ALU Src	Imm Src(1:0)	Reg Write
LW	00000	000	01	0	1	00	1
ALUI	00100	001	00	0	1	00	1
SW	01000	010	00	1	1	01	0
ALU	01100	011	00	0	0	XX	1
Branch	11000	100	00	0	0	10	0
JALR	11001	101	11	0	1	00	1
LUI	01101	11X	10	0	X	11	1

Τα σήματα **MemWrite** και **RegWrite** έχουν την τιμή 0 όταν δεν χρησιμοποιούνται η μνήμη δεδομένων και το αρχείο καταχωρητών

Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και τα πεδία **funct3** (*Instr*(14:12) και **funct7(5)** (*Instr*(30) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl(3:0)** και **SLTUorSLT**



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:0)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (3:0)	SLTUorSLT
ADD	01100	011	000	0	add	0000	X
SUB	01100	011	000	1	sub	0001	X
AND	01100	011	111	0	and	0100	X
OR	01100	011	110	0	or	0101	X
XOR	01100	011	100	0	xor	011X	X
SLL	01100	011	001	0	sll	1X0X	X
SRL	01100	011	101	0	srl	1X10	X
SRA	01100	011	101	1	sra	1X11	X
SLT	01100	011	010	0	slt	001X	0
SLTU	01100	011	011	0	sltu	001X	1

Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

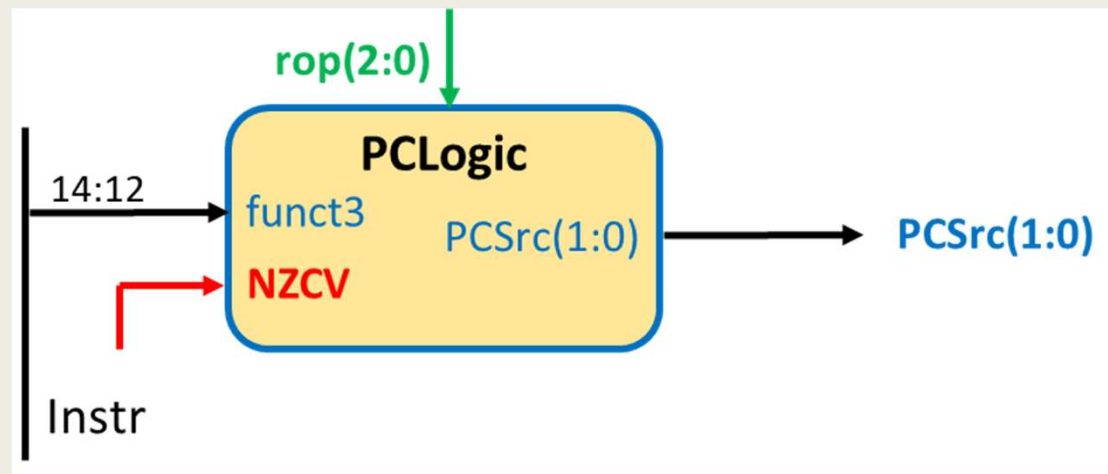
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:2)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (3:0)	SLTUorSLT
ADDI	00100	001	000	X	add	0000	X
ANDI	00100	001	111	X	and	0100	X
ORI	00100	001	110	X	or	0101	X
XORI	00100	001	100	X	xor	011X	X
LW	00000	000	010	X	add	0000	X
SW	01000	010	010	X	add	0000	X
BEQ	11000	100	000	X	sub	0001	X
BNE	11000	100	001	X	sub	0001	X
JALR	11001	101	000	X	add	0000	X
LUI	01101	11X	XXX	X	-	XXXX	X

Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και το πεδίο **funct3** (*Instr*(14:12), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές *Branch* με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των *BEQ/BNE*), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc(1:0)**



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

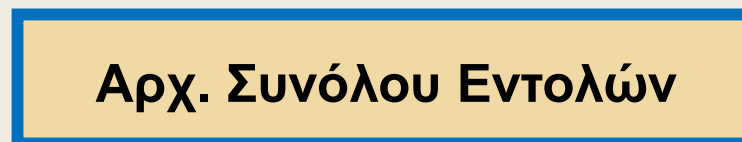
Εντολή	op(6:2)	rop(2:0)	funct3	NZCV	PCSrc(1:0)
LW	00000	000	XXX	XXXX	00
ALUI	00100	001	XXX	XXXX	00
SW	01000	010	XXX	XXXX	00
ALU	01100	011	XXX	XXXX	00
BEQ	11000	100	000	X0XX	00
BEQ	11000	100	000	X1XX	10
BNE	11000	100	001	X0XX	10
BNE	11000	100	001	X1XX	00
JALR	11001	101	XXX	XXXX	11
LUI	01101	11X	XXX	XXXX	00

Ανάλυση επιδόσεων επεξεργαστών

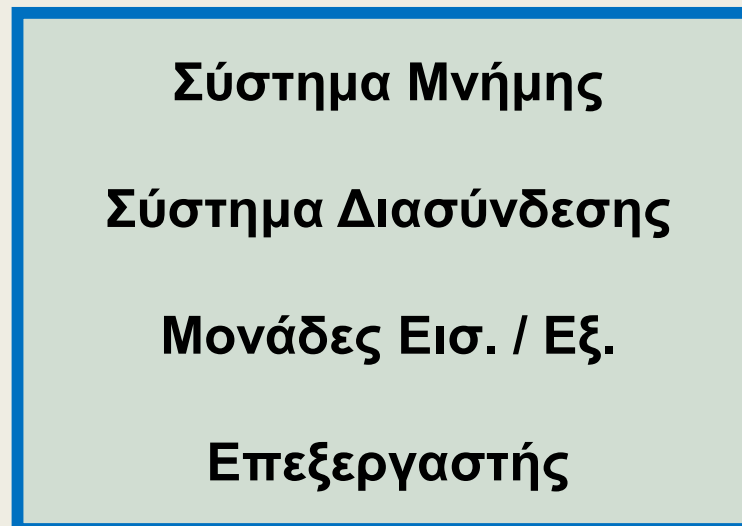
■ Τρόποι μέτρησης των επιδόσεων



Απαντήσεις ανά μήνα
Χρήσιμες λειτουργίες ανά δευτ/πτο



Millions of Instructions Per Sec. (MIPS)
Millions of Floating Point Operations
per Sec. (MFLOP/s)



Μεταφορά πληροφορίας (MBytes/s)
Χρόνος εκτέλεσης προγράμματος (s)
Συχνότητα λειτουργίας (MHz)

Ανάλυση επιδόσεων επεξεργαστών

- Ο μόνος τρόπος μέτρησης των επιδόσεων χωρίς τεχνάσματα εντυπωσιασμού είναι με βάση τον **χρόνο εκτέλεσης του προγράμματος (CPU_{time})** που μας ενδιαφέρει
 - Η αμέσως επόμενη καλύτερη επιλογή είναι να μετρήσουμε το συνολικό χρόνο εκτέλεσης μιας **συλλογής προγραμμάτων** (μετροπρογραμμάτων - *benchmarks*) που είναι παρόμοια με εκείνα που εκτελούμε
- Ο χρόνος εκτέλεσης ενός προγράμματος (σε δευτερόλεπτα) ορίζεται ως:
 - **$CPU_{time} = (\# \text{ instructions/program}) \times CPI \times T_c = \text{seconds/program}$**
 - πλήθος εντολών (# instructions/program)
 - μέσος αριθμός κύκλων ανά εντολή (average cycles per instruction – CPI)
 - περίοδος σήματος CLK (seconds/cycle – T_c)
- Στόχοι κατά την σχεδίαση ενός επεξεργαστή:
 - ελαχιστοποίηση χρόνου εκτέλεσης
 - ικανοποίηση περιορισμών που αφορούν: κόστος υλοποίησης και κατανάλωση ισχύος
- Αν και οι συνολικές επιδόσεις ενός υπολογιστή επηρεάζονται από πολλούς παράγοντες, θα εστιάσουμε μόνο στις επιδόσεις του επεξεργαστή

Ανάλυση επιδόσεων επεξεργαστών

- Πώς επηρεάζεται η επίδοση

- $\text{CPU}_{\text{time}} = (\# \text{ instructions/program}) \times \text{CPI} \times T_c = \text{seconds/program}$

	Πλήθος Εντολών	CPI	Περίοδος CLK
πρόγραμμα	X		
μεταγλωττιστής	X	X	
αρχ. συν. εντολών	X	X	X
οργάνωση		X	X
τεχνολογία			X

Η επίδοση του επεξεργαστή διαφοροποιείται από πρόγραμμα σε πρόγραμμα

Ανάλυση επιδόσεων επεξεργαστών

■ Πώς επηρεάζεται η επίδοση

- $\text{CPU}_{\text{time}} = (\# \text{ instructions/program}) \times \text{CPI} \times T_c = \text{seconds/program}$

■ Μέσος αριθμός κύκλων ανά εντολή (cycles per instruction - CPI):

- μέσος αριθμός κύκλων ανά εντολή του ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ενός προγράμματος
- εξαρτάται από το μίγμα των εντολών σε ένα πρόγραμμα (πόσες εντολές είναι επεξεργασίας δεδομένων, μνήμης ή διακλάδωσης)

■ Εντολές ανά κύκλο (instructions per cycle - IPC):

- διεκπεραιωτική ικανότητα - το αντίστροφο του CPI

■ Περίοδος T_c του ρολογιού (clock period):

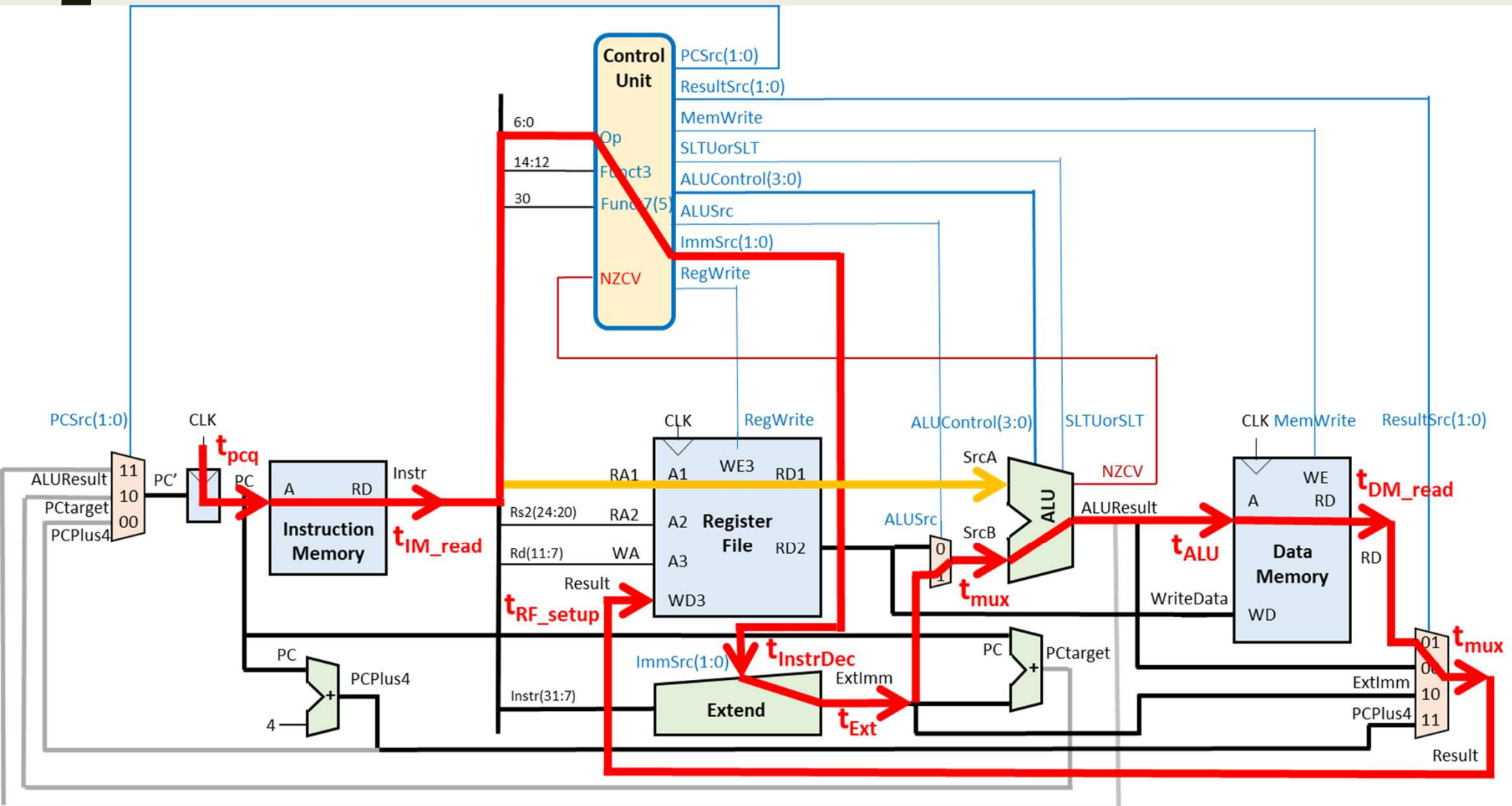
- το πλήθος των νανοδευτερολέπτων ανά κύκλο ρολογιού
- καθορίζεται από την **κρίσιμη διαδρομή (critical path)** μέσω της λογικής της εκάστοτε μικροαρχιτεκτονικής του επεξεργαστή
- ο τρόπος σχεδίασης της διαδρομής δεδομένων και των μονάδων που την απαρτίζουν επηρεάζουν την **κρίσιμη διαδρομή (critical path)** και συνεπώς την περίοδο του ρολογιού

Επεξεργαστής ενός κύκλου: ανάλυση επιδόσεων

- Ο επεξεργαστής ενός κύκλου έχει πάντα $CPI = 1$
- Η περίοδος του CLK καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση της **πιο αργής εντολής** μέσω της λογικής της μικροαρχιτεκτονικής ενός κύκλου
 - η πιο αργή εντολή είναι η **εντολή φόρτωσης LW**
- Λειτουργίες της εντολής LW με την αντίστοιχη καθυστέρηση διάδοσης που συμβάλλουν στην κρίσιμη διαδρομή:
 - φόρτωση νέας διεύθυνσης εντολής στον μετρητή PC (t_{pcq})
 - ανάγνωση νέας εντολής από τη μνήμη εντολών IM ως Instr (t_{IM_read})
 - υπολογισμός του ImmSrc(1:0) στον αποκωδ. εντολής InstrDec ($t_{InstrDec}$)
 - επιλογή επέκτασης πρόσήμου (ExtImm) στη μονάδα Extend (t_{Ext})
 - επιλογή του ExtImm ως SrcB της ALU μέσω πολυπλέκτη 2 σε 1 ($t_{mux2:1}$)
 - πρόσθεση SrcA με SrcB στη μονάδα ALU - το άθροισμα ως ALUResult (t_{ALU})
 - ανάγνωση τελεστέου από τη μνήμη δεδομένων DM ως ReadData (t_{DM_read})
 - επιλογή του ReadData (RD) ως Result μέσω πολυπλέκτη 4 σε 1 ($t_{mux4:1}$)
 - σταθεροποίηση Result για εγγραφή στο αρχείο καταχωρητών (t_{RF_setup})

Επεξεργαστής ενός κύκλου: ανάλυση επιδόσεων

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση της **εντολής LDR**
 - $T_c = t_{pcq} + t_{IM_read} + \max(t_{RF_read}, (t_{InstrDec} + t_{Ext} + t_{mux2:1})) + t_{ALU} + t_{DM_read} + t_{mux4:1} + t_{RF_setup}$



Επεξεργαστής ενός κύκλου: ανάλυση επιδόσεων

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση της **εντολής LW**
 - $T_c = t_{pcq} + t_{IM_read} + \max(t_{RF_read}, (t_{InstrDec} + t_{Ext} + t_{mux2:1})) + t_{ALU} + t_{DM_read} + t_{mux4:1} + t_{RF_setup}$
- Παράδειγμα υπολογισμού επιδόσεων επεξεργαστή ενός κύκλου:
 - η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:
 - $T_c = (40 + 200 + 100 + 150 + 200 + 30 + 60) \text{ ps} = 780 \text{ ps}$
 - ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:
 - $CPU_{time} = (\# \text{ instr/program}) \times CPI \times T_c = 100 \times 10^9 \times 1 \times 780 \times 10^{-12} = 78 \text{ s}$

Ενδεικτικές τιμές σε τεχνολογία CMOS 7-nm. Υποθέτουμε $t_{RF_read} > t_{InstrDec} + t_{Ext} + t_{mux2:1}$

Στοιχεία	Παράμετρος	Καθυστερήση (ps)
Καθυστερήση από το CLK έως την έξοδο Q (καταχωρητής)	T_{pcq}	40
Χρόνος σταθεροποίησης καταχωρητή	T_{setup}	50
Χρόνος σταθεροποίησης αρχείου καταχωρητών	t_{RF_setup}	60
Καθ. διάδ. στην μονάδα ελέγχου InstrDec/ALUDec/PCLogic	$t_{InD}/t_{ALUD}/t_{PCLogic}$	25/35/35(10 από NZCV)
Καθυστερήση διάδοσης (control path) στη μονάδα Extend	t_{Ext}	35
Καθυστερήση διάδοσης πολυπλέκτη 2σε1/4σε1/8σε1	$T_{mux2:1/4:1/8:1}$	20/30/50
Καθυστερήση διάδοσης στη μονάδα ALU/στον αθροιστή	t_{ALU}/t_{adder}	150/100
Χρόνος ανάγνωσης από τη μνήμη εντολών (ROM)	t_{IM_read}	200
Χρόνος ανάγνωσης από τη μνήμη δεδομένων (RAM)	t_{DM_read}	200
Χρόνος ανάγνωσης από το αρχείο καταχωρητών (RAM)	t_{RF_read}	100

Επεξεργαστής ενός κύκλου: Αδυναμίες

- Απαιτεί έναν κύκλο ρολογιού τόσο μεγάλο ώστε να υποστηρίζεται η πιο αργή εντολή (LW)
 - παρά το γεγονός ότι οι υπόλοιπες εντολές θα μπορούσαν να εκτελούνται πιο γρήγορα
 - αναγκαιότητα χρήσης *μη αρχιτεκτονικών καταχωρητών* ανάμεσα σε τμήματα συνδυαστικής λογικής, ώστε να διαιρεθεί η λογική σε μικρότερα στάδια, τα οποία μπορούν να εκτελούνται με γρηγορότερο ρολόι
 - Η περίοδος του ρολογιού εξαρτάται από το πιο αργό στάδιο
- Απαιτεί ξεχωριστές μνήμες για τις εντολές και τα δεδομένα
 - οι περισσότεροι απλοί επεξεργαστές διαθέτουν μόνο μία εξωτερική μνήμη στην οποία αποθηκεύονται και οι εντολές και τα δεδομένα
 - στο χαμηλότερο ιεραρχικά επίπεδο μνήμης (cache L1) οι μνήμες εντολών και δεδομένων είναι ξεχωριστές στην πράξη
- Απαιτεί τρεις αθροιστές: έναν στη μονάδα ALU και δύο για τη λογική του μετρητή PC (PCplus4 και Pctarget)
 - οι αθροιστές είναι σχετικά ακριβά κυκλώματα, ειδικά αν πρέπει να είναι γρήγοροι (π.χ. αθροιστές πρόβλεψης κρατούμενου)
 - δυνατότητα αντικατάστασης με πολυπλέκτες, η οποία αυξάνει όμως την πολυπλοκότητα στη ροή και τον έλεγχο των δεδομένων

Στην συνέχεια παρουσιάζουμε τους επεξεργαστές πολλών κύκλων και διοχέτευσης με εισαγωγή ενδιάμεσων καταχωρητών που αυξάνουν την συχνότητα του ρολογιού

Επεξεργαστής πολλών κύκλων

- Η σχεδίαση της **μικροαρχιτεκτονικής πολλών κύκλων** γίνεται με βάση την πλήρη μικροαρχιτεκτονική ενός κύκλου που υλοποιεί **όλες τις βασικές εντολές**
- Σχεδίαση της διαδρομής δεδομένων με προσθήκη **μη αρχιτεκτονικών καταχωρητών**, ώστε να εκτελείται η εντολή με πολλούς κύκλους σε 5 βήματα, στοχεύοντας:
 - στην σημαντική **μείωση της περιόδου του ρολογιού**
 - που καθορίζεται από την **κρίσιμη διαδρομή (critical path)** του πιο αργού από τα 5 βήματα εκτέλεσης της εντολής
 - σε **διαφορετικό** πλήθος κύκλων εκτέλεσης (από 3 μέχρι 5 κύκλους) ανά εντολή
 - κύκλοι εκτέλεσης βημάτων που δεν απαιτούνται, παραλείπονται
 - δυνατότητα εκτέλεσης περισσότερων βημάτων παράλληλα στον ίδιο κύκλο
- Σχεδίαση της μονάδας ελέγχου κατά τέτοιο τρόπο, ώστε:
 - τα σήματα **έγκρισης εγγραφής** να παράγονται από **μηχανή πεπερασμένων καταστάσεων**, ώστε να έχουν την σωστή τιμή 0 ή 1 σε κάθε κύκλο εκτέλεσης της εντολής
 - τα υπόλοιπα σήματα ελέγχου που παραμένουν **σταθερά σε όλους τους κύκλους** κατά την εκτέλεση της εντολής να παράγονται από **συνδυαστική λογική**
- Απαιτείται η προσθήκη ενός επιπλέον **σήματος έγκρισης εγγραφής στον μετρητή προγράμματος PC (PCWrite)**,
 - που ενεργοποιείται στο τελευταίο 5^ο βήμα της εκτέλεσης της εντολής, ώστε να εγγράφεται στον μετρητή προγράμματος PC η διεύθυνση της επόμενης προς εκτέλεση εντολής στην αμέσως επόμενη ανερχόμενη ακμή του ρολογιού
 - σε ποιον κύκλο εκτελείται το τελευταίο 5^ο στάδιο ποικίλει από εντολή σε εντολή.
 - Ενδεικτικά, στην εντολή LW εκτελείται στον 5^ο κύκλο, στις εντολές SW/ADD/XORI/SLL στον 4^ο κύκλο και στις εντολές BEQ/JALR/LUI στον 3^ο κύκλο (με κατάλληλη σχεδίαση)

Επεξεργαστής πολλών κύκλων

- Αρχικά, οι λειτουργίες των εντολών ταξινομούνται σε πέντε βήματα εκτέλεσης της εντολής, που απορρέουν από την μελέτη των στοιχείων που έχουν τις μεγαλύτερες καθυστερήσεις διάδοσης/χρόνους ανάγνωσης και συνεισφέρουν σημαντικά στον χρόνο εκτέλεσης της εντολής
 - Μονάδα ALU (150 ps)
 - Αθροιστές - CLA (100 ps)
 - Μνήμη εντολών (ROM) (200 ps)
 - Μνήμη δεδομένων (RAM) (200 ps)
 - Αρχείο καταχωρητών (RAM) (100 ps)

Ενδεικτικές τιμές σε τεχνολογία CMOS 7-nm.

Στοιχεία	Παράμετρος	Καθυστέρηση (ps)
Καθυστέρηση από το CLK έως την έξοδο Q (καταχωρητής)	t_{pcq}	40
Χρόνος σταθεροποίησης καταχωρητή	t_{setup}	50
Χρόνος σταθεροποίησης αρχείου καταχωρητών	t_{RF_setup}	60
Καθ. διάδ. στην μονάδα ελέγχου InstrDec/ALUDec/PCLogic	$t_{InD}/t_{ALUD}/t_{PClogic}$	25/35/35(10 από NZCV)
Καθυστέρηση διάδοσης (control path) στη μονάδα Extend	t_{Ext}	35
Καθυστέρηση διάδοσης πολυπλέκτη 2σε1/4σε1/8σε1	$T_{mux2:1/4:1/8:1}$	20/30/50
Καθυστέρηση διάδοσης στη μονάδα ALU/στον αθροιστή	t_{ALU}/t_{adder}	150/100
Χρόνος ανάγνωσης από τη μνήμη εντολών (ROM)	t_{IM_read}	200
Χρόνος ανάγνωσης από τη μνήμη δεδομένων (RAM)	t_{DM_read}	200
Χρόνος ανάγνωσης από το αρχείο καταχωρητών (RAM)	t_{RF_read}	100

Επεξεργαστής πολλών κύκλων

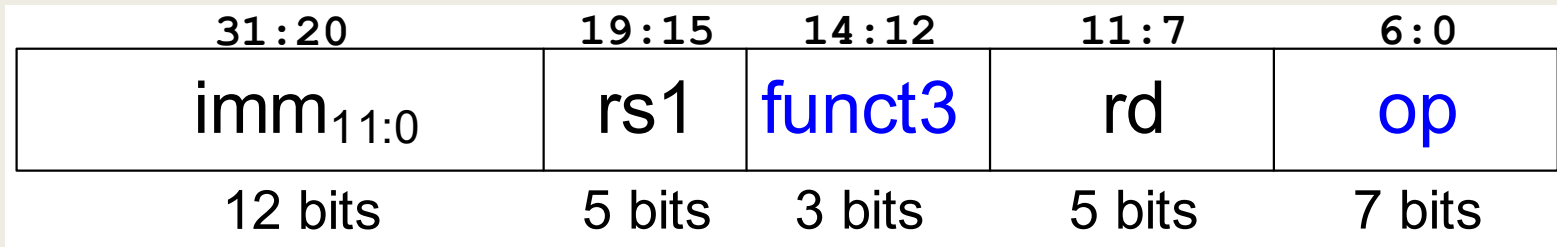
- Μετά την ταξινόμηση των λειτουργιών σε 5 διακριτά βήματα εκτέλεσης της εντολής, όπου κάθε ένα βήμα υλοποιείται χωριστά, προκύπτουν τα ακόλουθα βήματα:
 1. Προσκόμιση εντολής και υπολογισμός της διεύθυνσης της επόμενης εντολής (PC+4)
 2. Αποκωδικοποίηση εντολής, ανάγνωση αρχείου καταχωρητών και υπολογισμός άμεσου τελεστέου των 32 bit
 3. Εκτέλεση πράξεων στη μονάδα ALU και στον αθροιστή PCtarget
 4. Ανάγνωση ή εγγραφή στη μνήμη δεδομένων
 5. Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών και επιλογή της διεύθυνσης της επόμενης εντολής
- Συνεκτιμώντας ότι η υλοποίηση θα γίνει σε τεχνολογία FPGA προβαίνουμε στις ακόλουθες αρχιτεκτονικές επιλογές:
 - η **μνήμη εντολών** παραμένει **χωριστά** από τη **μνήμη δεδομένων**, ώστε να υλοποιηθούν ανεξάρτητα ως μνήμες ROM και RAM, αντίστοιχα
 - διατηρούνται οι **δύο αθροιστές PCplus4** και **PCtarget**, αξιοποιώντας τους διαθέσιμους πόρους (CARRY4) του FPGA χωρίς ιδιαίτερη επιβάρυνση σε LUTs
 - ανάμεσα στα υλοποιημένα βήματα τοποθετούνται **μη αρχιτεκτονικοί καταχωρητές**, αξιοποιώντας διαθέσιμα D-Flip-flops

Επεξεργαστής πολλών κύκλων

- Βασικές εντολές που υλοποιούνται:

- Εντολή *LW* τύπου *I* (*immediate*):

- `LW      Rd, imm(Rs1)      Rd = mem[Rs1+S_Ext(imm)]`

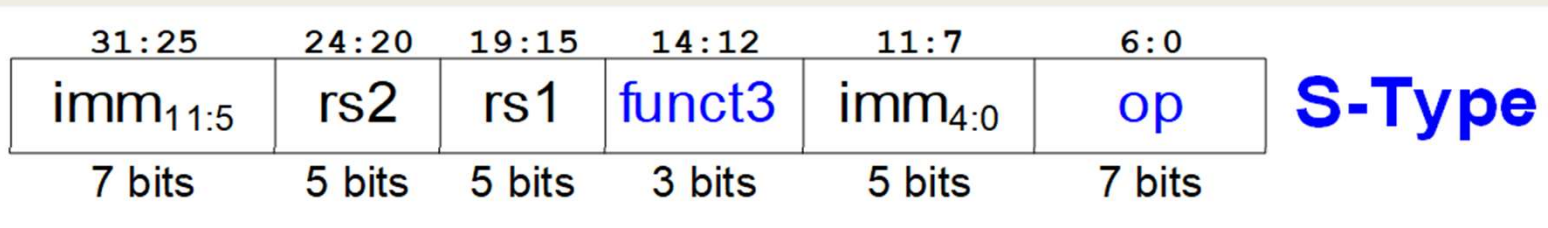


Επεξεργαστής πολλών κύκλων

- Βασικές εντολές που υλοποιούνται:

- *Εντολή SW τύπου S (store):*

- `SW Rs2, imm(Rs1)      mem[Rs1+S_Ext(imm)] = Rs2`



Επεξεργαστής πολλών κύκλων

■ Βασικές εντολές που υλοποιούνται:

– Εντολές ALU τύπου R (register):

■	ADD	Rd, Rs1, Rs2	$Rd = Rs1 + Rs2$
■	SUB	Rd, Rs1, Rs2	$Rd = Rs1 - Rs2$
■	AND	Rd, Rs1, Rs2	$Rd = Rs1 \text{ and } Rs2$
■	OR	Rd, Rs1, Rs2	$Rd = Rs1 \text{ or } Rs2$
■	XOR	Rd, Rs1, Rs2	$Rd = Rs1 \text{ xor } Rs2$
■	SLT	Rd, Rs1, Rs2	$Rd = 1 \text{ if } rs1 < rs2, \text{ else } 0$
■	SLTU	Rd, Rs1, Rs2	$Rd = 1 \text{ if } rs1 < rs2, \text{ else } 0$
■	SLL	Rd, Rs1, Rs2	$Rd = Rs1 \ll Rs2(4:0)$
■	SRL	Rd, Rs1, Rs2	$Rd = Rs1 \gg Rs2(4:0)$
■	SRA	Rd, Rs1, Rs2	$Rd = Rs1 \ggg Rs2(4:0)$

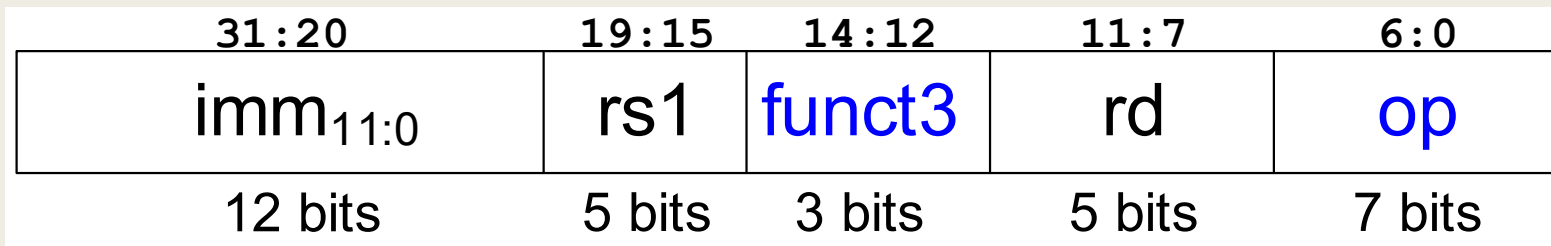
31:25	24:20	19:15	14:12	11:7	6:0
funct7	rs2	rs1	funct3	rd	op
7 bits	5 bits	5 bits	3 bits	5 bits	7 bits

Επεξεργαστής πολλών κύκλων

- Βασικές εντολές που υλοποιούνται:

- *Εντολές ALU τύπου I (immediate):*

- `ADDI Rd, Rs1, imm` $Rd = Rs1 + S_Ext(imm)$
 - `ANDI Rd, Rs1, imm` $Rd = Rs1 \text{ and } S_Ext(imm)$
 - `ORI Rd, Rs1, imm` $Rd = Rs1 \text{ or } S_Ext(imm)$
 - `XORI Rd, Rs1, imm` $Rd = Rs1 \text{ xor } S_Ext(imm)$



Επεξεργαστής πολλών κύκλων

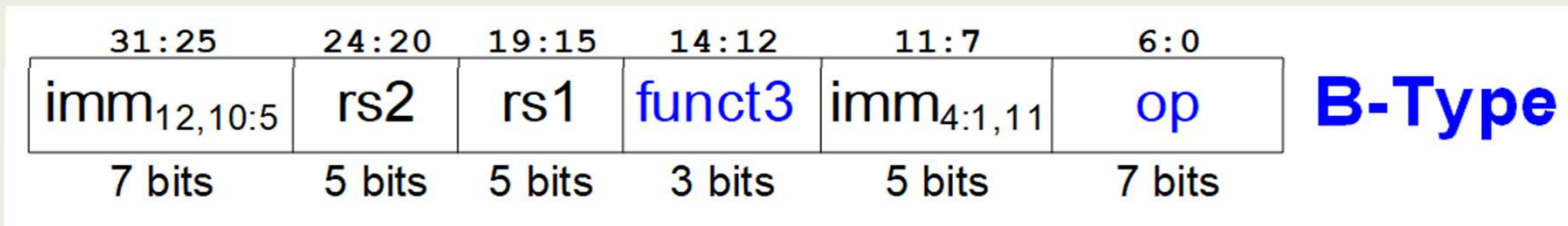
- Βασικές εντολές που υλοποιούνται:

- *Εντολές τύπου B (branch):*

- `BEQ Rs1, Rs2, label branch` BTA if `Rs1 = Rs2`

- `BNE Rs1, Rs2, label branch` BTA if `Rs1 ≠ Rs2`

`BTA = PC + Sign_Ext(instr(31)& instr(7)&instr(30:25)&instr(11:8)&'0')`



Επεξεργαστής πολλών κύκλων

- Βασικές εντολές που υλοποιούνται:

- Εντολή *JALR* τύπου *I* (*immediate*):

- `JALR Rd, Rs1, imm` $PC = Rs1 + S_Ext(imm)$
 $Rd = PC + 4$

`JR ra -> JALR zero, ra, 0` $PC = ra + 0$
 $zero = PC + 4$

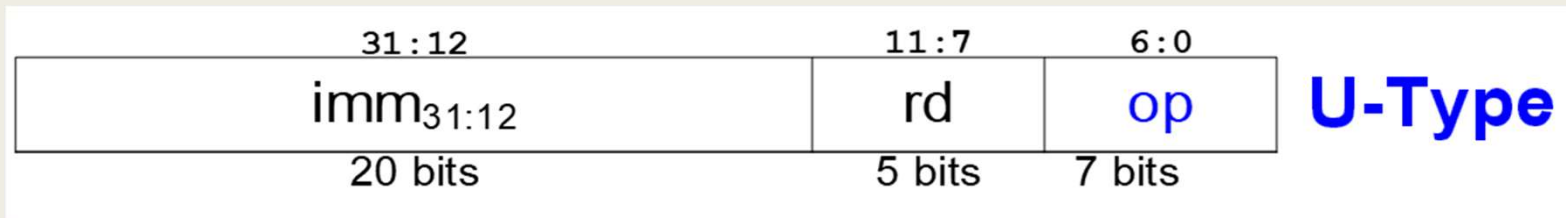
31:20	19:15	14:12	11:7	6:0
imm _{11:0}	rs1	funct3	rd	op
12 bits	5 bits	3 bits	5 bits	7 bits

Επεξεργαστής πολλών κύκλων

- Βασικές εντολές που υλοποιούνται::

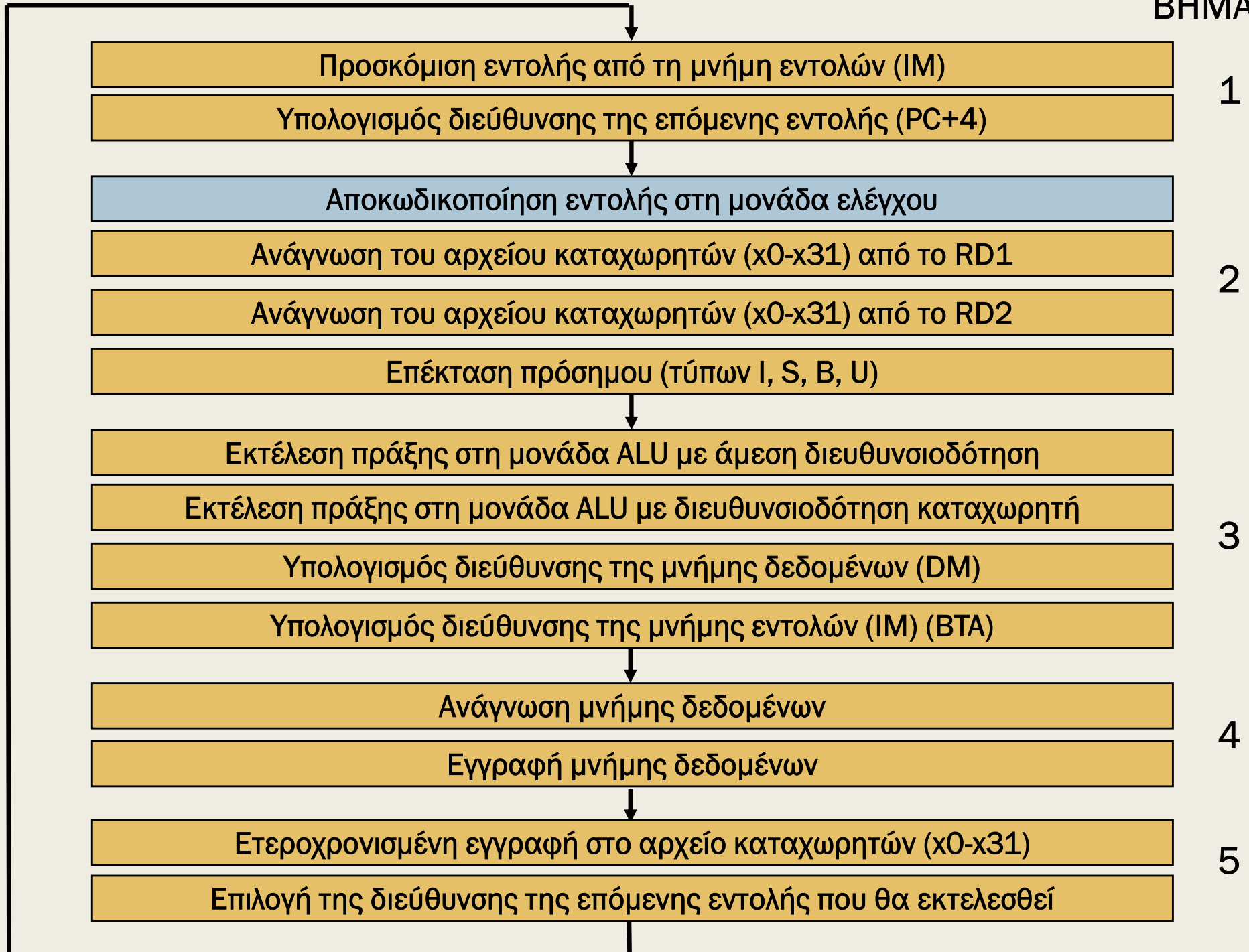
- Εντολές τύπου U (*upper immediate*):

- LUI Rd, upimm Rd = (upimm, 0x000)



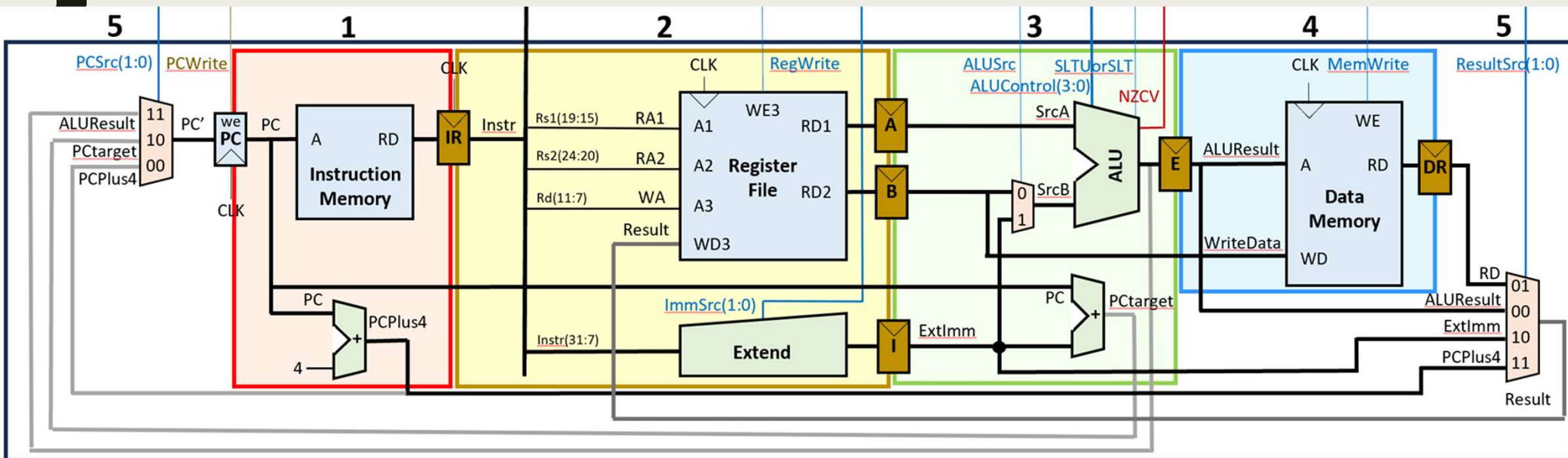
Λειτουργίες των βασικών εντολών (14)

ΒΗΜΑΤΑ



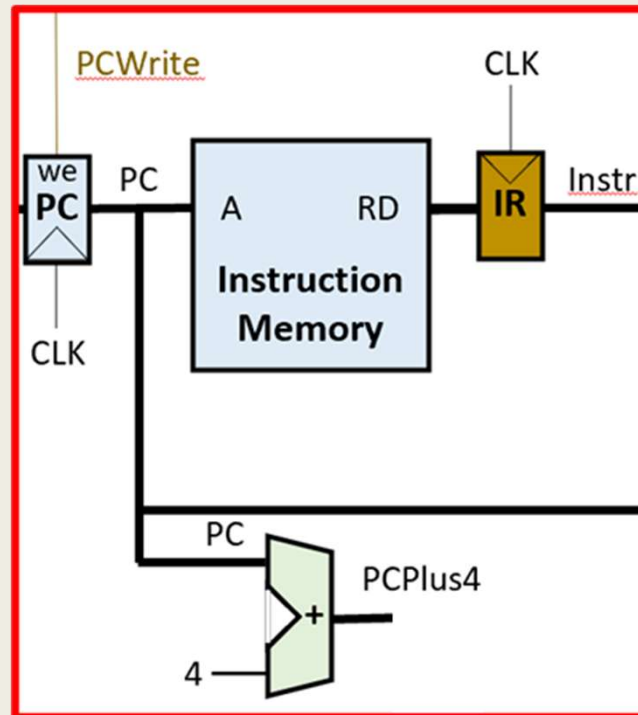
Επεξεργαστής πολλών κύκλων

- Κατά τη σχεδίαση προβαίνουμε σε:
 - μελέτη διαδρομής δεδομένων ανά βήμα εκτέλεσης της εντολής
 - μελέτη μονάδας ελέγχου
 - ανάλυση επιδόσεων
- Μετά την μελέτη της διαδρομής δεδομένων και την τοποθέτηση των απαιτούμενων μη αρχιτεκτονικών καταχωρητών η διαδρομή δεδομένων σε 5 βήματα του επεξεργαστή πολλών κυκλών είναι η ακόλουθη:



Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Αρχικά, τοποθετούμε τους μη αρχιτεκτονικούς καταχωρητές που αφορούν στο βήμα 1 «Προσκόμιση εντολής και υπολογισμός της διεύθυνσης PC+4»:
 - *Instruction Register (IR)* στη θύρα ανάγνωσης (A/RD) της μνήμης εντολών χωρίς σήμα έγκρισης εγγραφής *IRWrite*
 - η τρέχουσα εντολή αλλάζει μόνο μετά το τελευταίο στάδιο υλοποίησής της

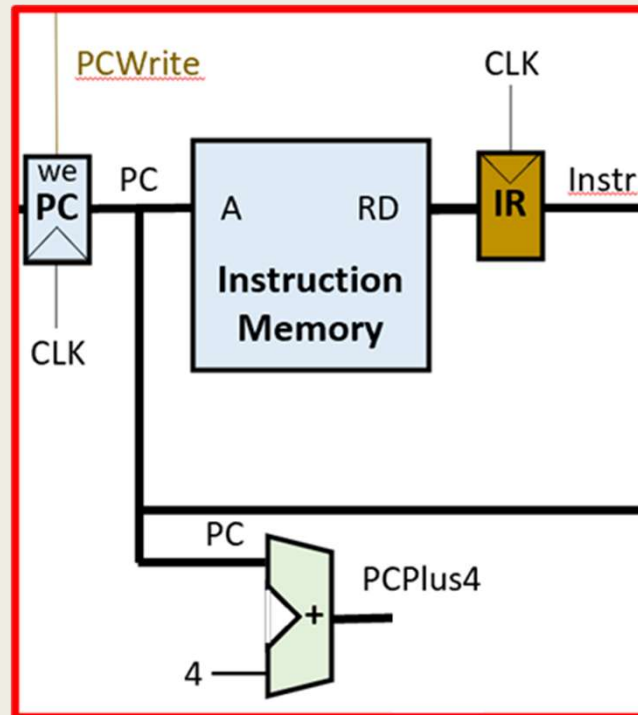


Ο **program counter (PC)** απαιτεί τη χρήση σήματος έγκρισης εγγραφής **PCWrite**, ώστε να διατηρεί το περιεχόμενο του σταθερό (**PCWrite=0**) μέχρι την ολοκλήρωση της εκτέλεσης της εντολής

Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Βήμα 1: Λειτουργίες που εκτελούνται (κοινές για όλες τις εντολές)
 - Προσκόμιση εντολής από την μνήμη εντολών (*Instruction Memory - IM*)
 - $IR = IM(PC)$ (απαιτείται $PCWrite = 0$)
 - Υπολογισμός διεύθυνσης της επόμενης εντολής ($PC+4$)
 - $PCplus4 = PC + 4$ (απαιτείται $PCWrite = 0$)

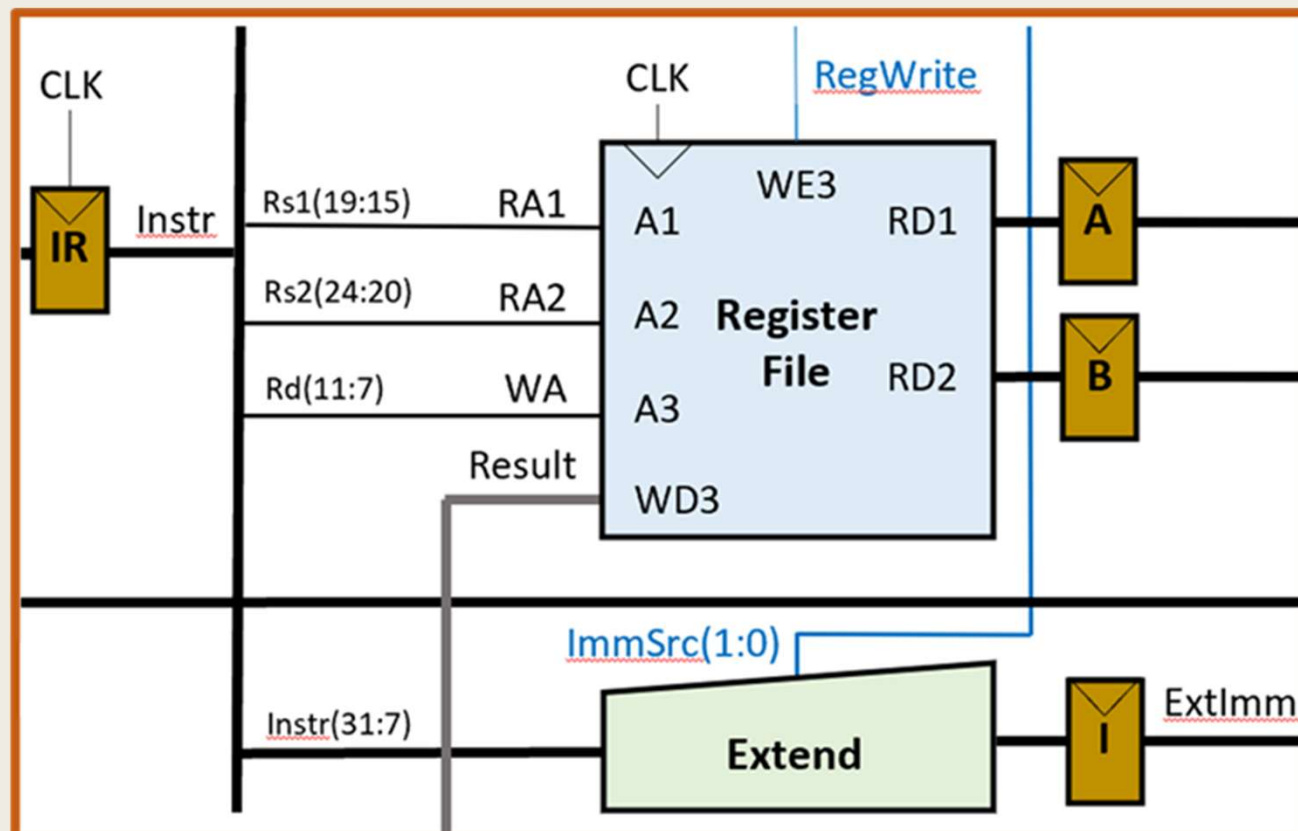


Ο **program counter (PC)** απαιτεί τη χρήση σήματος έγκρισης εγγραφής **PCWrite**, ώστε να διατηρεί το περιεχόμενο του σταθερό (**PCWrite=0**) μέχρι την ολοκλήρωση της εκτέλεσης της εντολής

Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

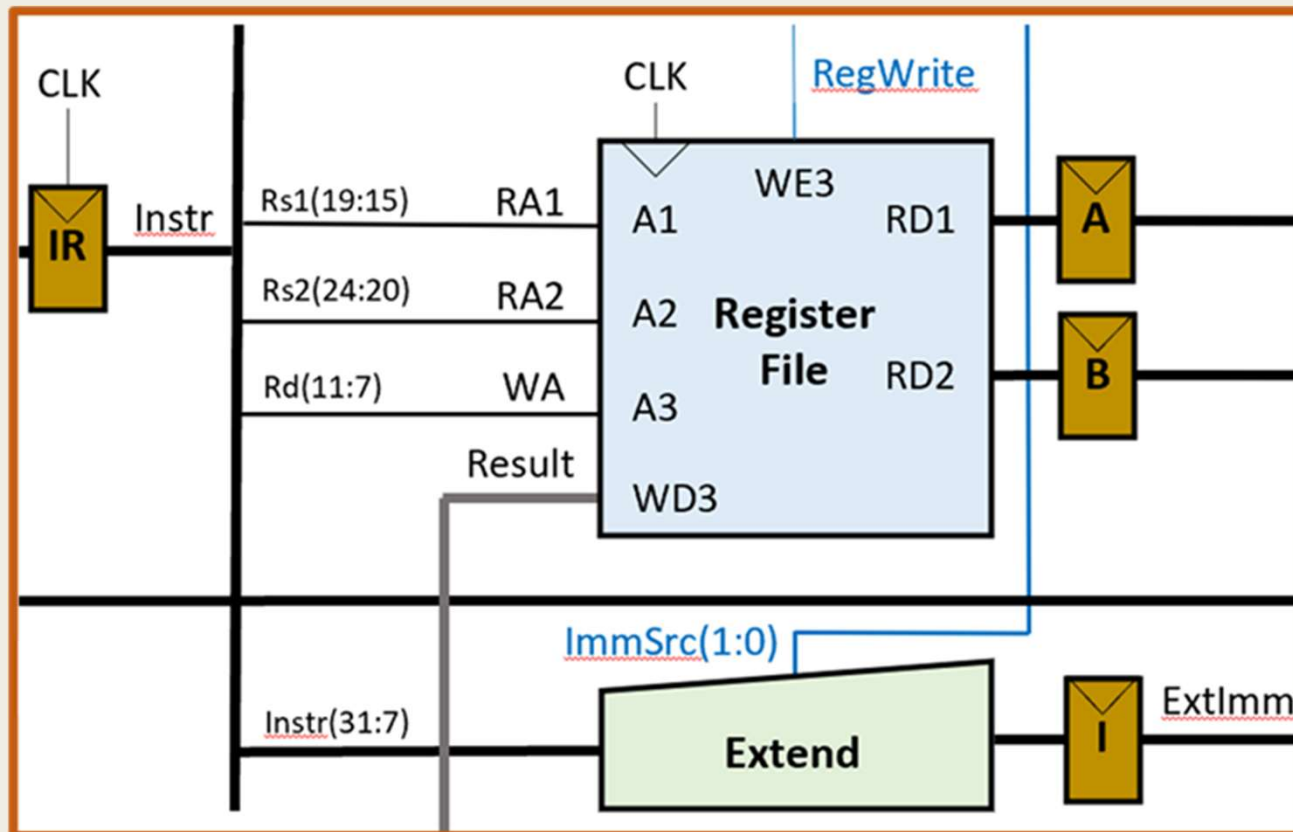
Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Στη συνέχεια, τοποθετούμε τους μη αρχιτεκτονικούς καταχωρητές που αφορούν στο βήμα 2 «Αποκωδικοποίηση εντολής, ανάγνωση αρχείου καταχωρητών και υπολογισμός άμεσου τελεστέου των 32 bit»:
 - **Register A (A)** στη θύρα ανάγνωσης (A1/RD1) του αρχείου καταχωρητών
 - **Register B (B)** στη θύρα ανάγνωσης (A2/RD2) του αρχείου καταχωρητών
 - **Register I (I)** στην έξοδο ExtImm της μονάδας Extend



Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Βήμα 2: Λειτουργίες που εκτελούνται (για όλες τις εντολές)
 - Ανάγνωση του αρχείου καταχωρητών (x0-x31) από τα RD1/RD2
 - $A = RF(Rs1)$ (απαιτείται PCWrite = 0 και RegWrite = 0)
 - $B = RF(Rs2)$ (απαιτείται PCWrite = 0 και RegWrite = 0)

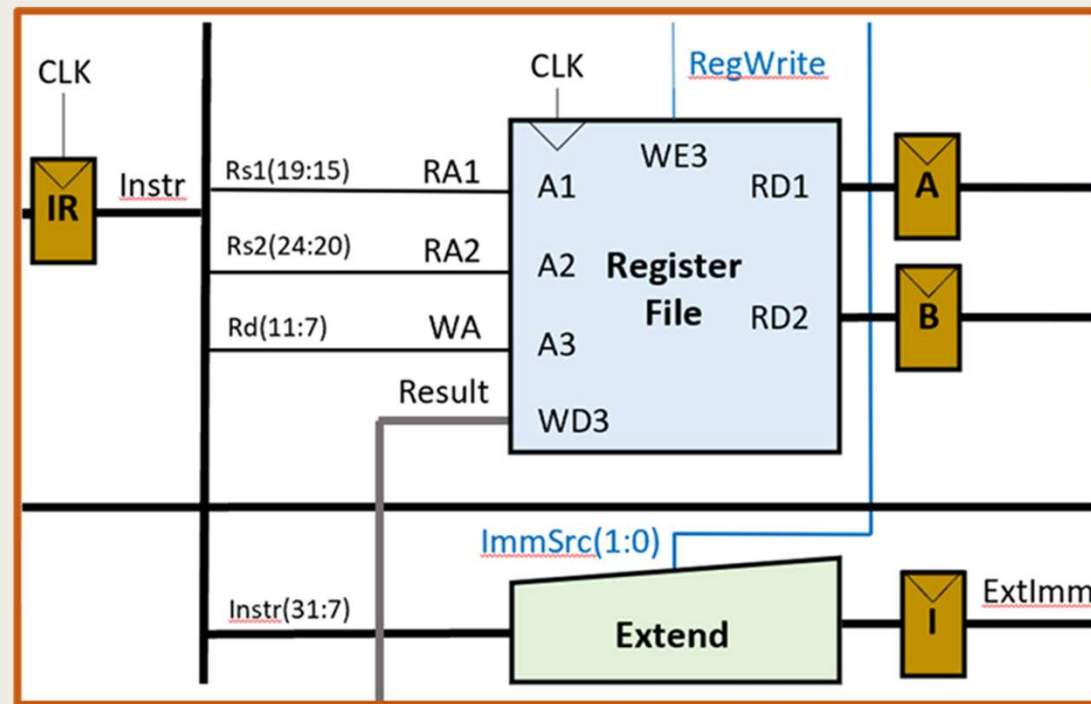


Ο καταχωρητής A έχει έγκυρη τιμή στις εντολές τύπου R, τύπου I και τύπου S/B

Ο καταχωρητής B έχει έγκυρη τιμή στις εντολές τύπου R και τύπου S/B

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

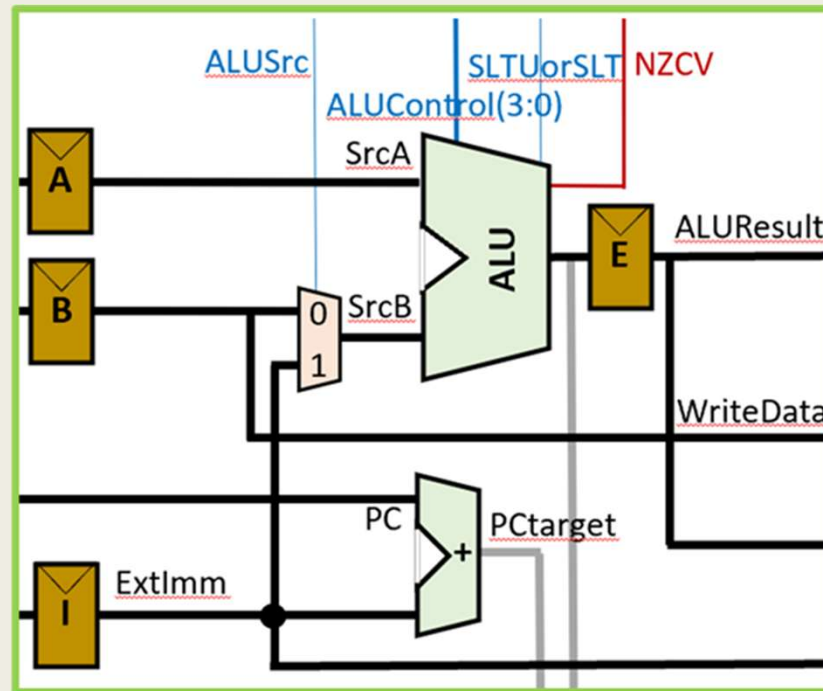
- Βήμα 2: Λειτουργίες που εκτελούνται (για όλες τις εντολές των τύπων I, S, B, U)
 - Επέκταση προσήμου άμεσου τελεστέου στα 32 bit (τύπων I, S, B, U, αντίστοιχα)
 - I: $I = \text{Sign_Ext}(\text{instr}(31:20))$ (απαιτείται PCWrite = 0 και ImmSrc(1:0) = 00)
 - S: $I = \text{Sign_Ext}(\text{instr}(31:25) \& \text{instr}(11:7))$ (απαιτείται PCWrite = 0 και ImmSrc(1:0) = 01)
 - B: $I = \text{Sign_Ext}(\text{instr}(31) \& \text{instr}(7) \& \text{instr}(30:25) \& \text{instr}(11:8) \& '0')$ (απαιτείται PCWrite = 0 και ImmSrc(1:0) = 10)
 - U: $I = \text{instr}(31:12) \& 0x000$ (απαιτείται PCWrite = 0 και ImmSrc(1:0) = 11)



11	10	9	8	7	6	5	4	3	2	1	0	rs1	funct3	rd										
11	10	9	8	7	6	5	rs2					rs1	funct3	4	3	2	1	0						
12	10	9	8	7	6	5	rs2					rs1	funct3	4	3	2	1	11						
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	rd				
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	rd			
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Στη συνέχεια, τοποθετούμε τους μη αρχιτεκτονικούς καταχωρητές που αφορούν στο βήμα 3 «Εκτέλεση πράξεων στη μονάδα ALU και στον αθροιστή PCtarget»:
 - **Register E (E)** στην έξοδο ALUResult της μονάδας ALU



Προσοχή! Προωθούνται απευθείας η έξοδος **PCtarget** του αθροιστή στον PC και το **Z (flag)** στην Μονάδα Ελέγχου για να είναι δυνατή η ολοκλήρωση των εντολών τύπου B στο βήμα 3.

Προσοχή! Προωθείται απευθείας η έξοδος **ALUResult** της μονάδας ALU στον PC για να είναι δυνατή η ολοκλήρωση των εντολών JALR/JR στο βήμα 3

Προσοχή! Προωθείται απευθείας η έξοδος **Extimm** του καταχωρητή I στην είσοδο WD του αρχείου καταχωρητών για να είναι δυνατή η ολοκλήρωση της εντολής LUI στο βήμα 3

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

■ Βήμα 3: Λειτουργίες που εκτελούνται (για όλες τις εντολές των τύπων R, I, S, B)

- Εκτέλεση πράξεων στη μονάδα ALU (ALUSrc=0)

- ADD $E = A + B$
- SUB $E = A - B$
- AND $E = A \text{ and } B$
- OR $E = A \text{ or } B$
- XOR $E = A \text{ xor } B$
- SLT $E = 1, \text{ if } A < B, \text{ else } 0$
- SLTU $E = 1, \text{ if } A < B, \text{ else } 0$
- SLL $E = A \ll B(4:0)$
- SRL $E = A \gg B(4:0)$
- SRA $E = A \ggg B(4:0)$
- BEQ/BNE $Z = 1, \text{ if } A=B, \text{ else } 0$

- Εκτέλεση πράξεων στη μονάδα ALU (ALUSrc=1)

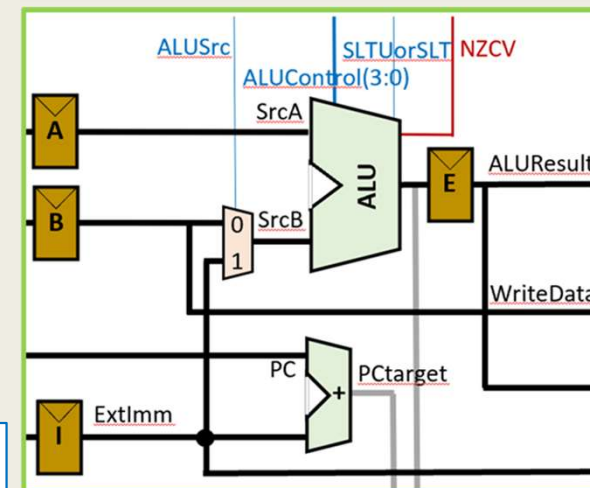
- LW $E = A + I$
- SW $E = A + I$
- ADDI $E = A + I$
- ANDI $E = A \text{ and } I$
- ORI $E = A \text{ or } I$
- XORI $E = A \text{ xor } I$
- JALR/JR $ALUResult = A + I$

- Εκτέλεση πράξεων στον αθροιστή PCtarget

- BEQ/BNE $PCtarget = PC + I \text{ (BTA)}$

BTA = PC + Sign_Ext(instr(31)&instr(7)&instr(30:25)&instr(11:8)&'0')

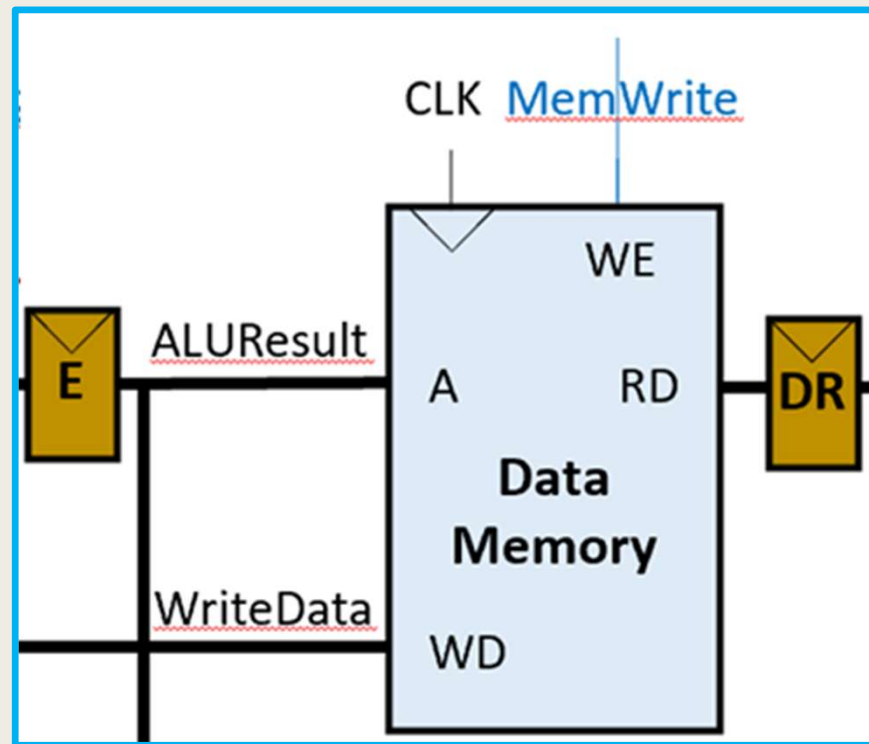
ALUControl(3:0)	Πράξη
0000	Add
0001	Subtract
0011 (SLTUorSLT=0)	SLT
0011 (SLTUorSLT=1)	SLTU
0100	AND
0101	OR
011X	XOR
1X0X	SLL
1X10	SRL
1X11	SRA



Τα σήματα **Z (flag)**, **ALUResult** και **PCtarget** δεν αποθηκεύονται σε μη αρχιτεκτονικούς καταχωρητές

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Στη συνέχεια, τοποθετούμε τους μη αρχιτεκτονικούς καταχωρητές που αφορούν στο βήμα 4 «Ανάγνωση ή εγγραφή στη μνήμη δεδομένων»:
 - **Memory Data Register Output (DR)** στην θύρα ανάγνωσης (A/RD) της μνήμης δεδομένων

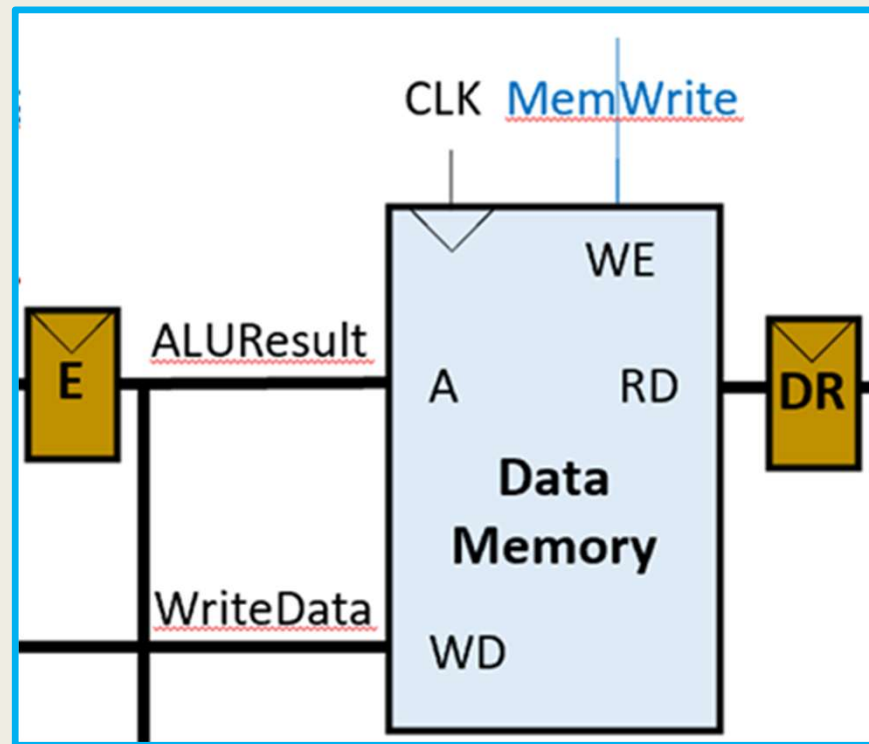


Ο καταχωρητής E χρησιμοποιείται ως Memory Address Register (MAR) χωρίς σήμα έγκρισης εγγραφής **MARWrite**. Δεν απαιτείται επιπλέον καταχωρητής MAR με σήμα έγκρισης εγγραφής **MARWrite**.

Η έξοδος **WriteData** του καταχωρητή B συνδέεται απευθείας στην είσοδο WD της μνήμης δεδομένων για εντολές τύπου S. Δεν απαιτείται επιπλέον καταχωρητής Memory Write Data Register.

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Στη συνέχεια, τοποθετούμε τους μη αρχιτεκτονικούς καταχωρητές που αφορούν στο βήμα 4 «Ανάγνωση ή εγγραφή στη μνήμη δεδομένων»:
 - *Memory Read Data Register (DR)* στην θύρα ανάγνωσης (A/RD) της μνήμης δεδομένων

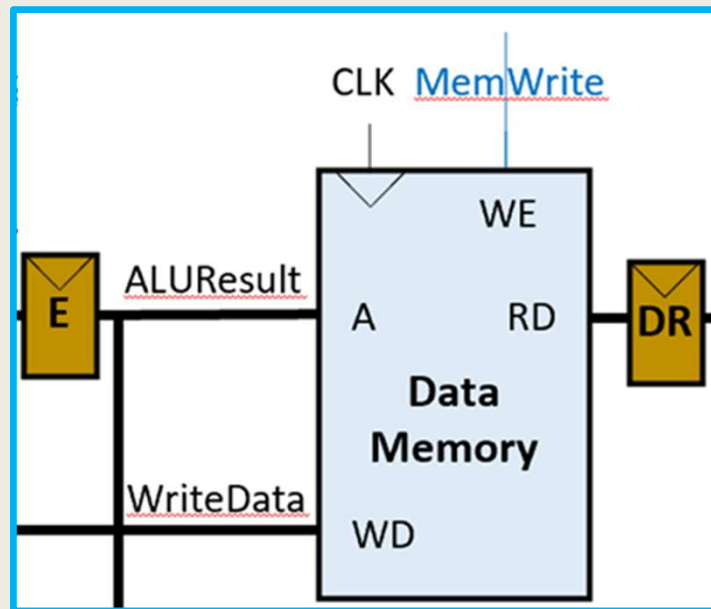


Ο καταχωρητής E χρησιμοποιείται ως Memory Address Register (MAR) χωρίς σήμα έγκρισης εγγραφής **MARWrite**. Δεν απαιτείται επιπλέον καταχωρητής MAR με σήμα έγκρισης εγγραφής **MARWrite**.

Η έξοδος **WriteData** του καταχωρητή B συνδέεται απευθείας στην είσοδο WD της μνήμης δεδομένων για εντολές τύπου S. Δεν απαιτείται επιπλέον καταχωρητής Memory Write Data Register.

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

- Βήμα 4: Λειτουργίες που εκτελούνται
 - Ανάγνωση από την μνήμη Δεδομένων (απαιτείται $MemWrite = 0$)
 - LW $DR = DM(E)$ (το DR έχει έγκυρη τιμή στο Βήμα 5)
 - Εγγραφή στην μνήμη Δεδομένων (απαιτείται $MemWrite = 1$)
 - SW $DM(E) = B$ (έγκυρη εγγραφή στο Βήμα 4)



Ο καταχωρητής E χρησιμοποιείται ως Memory Address Register (MAR) χωρίς σήμα έγκρισης εγγραφής **MARWrite**. Δεν απαιτείται επιπλέον καταχωρητής MAR με σήμα έγκρισης εγγραφής **MARWrite**.

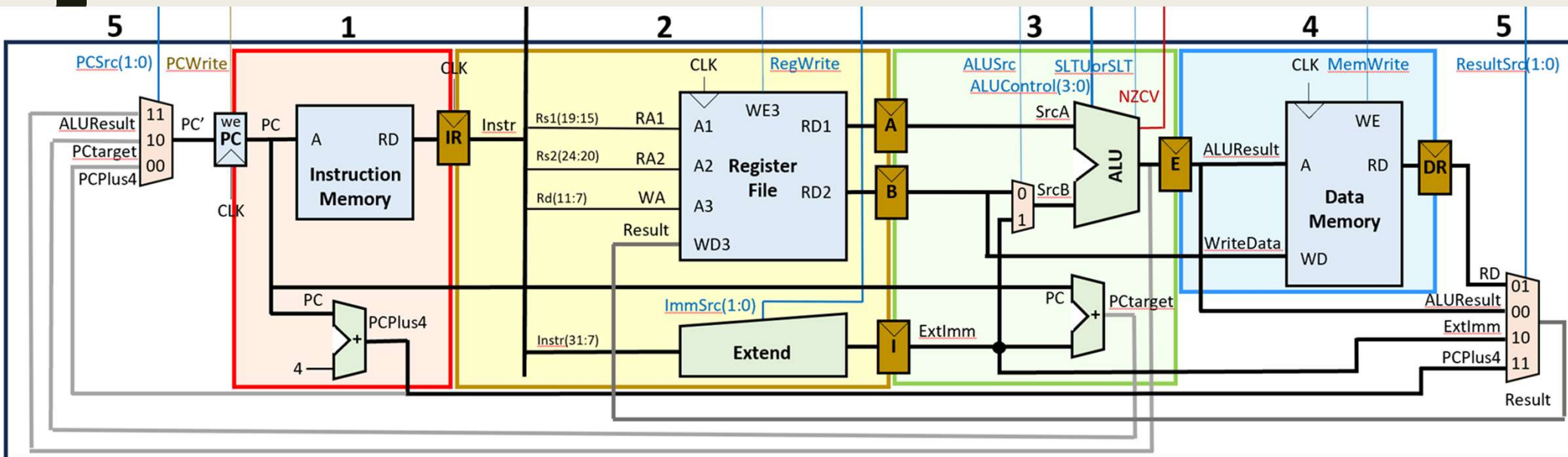
Η έξοδος **WriteData** του καταχωρητή B συνδέεται απευθείας στην είσοδο WD της μνήμης δεδομένων για εντολές τύπου S. Δεν απαιτείται επιπλέον καταχωρητής Memory Write Data Register

Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

■ Βήμα 5: Λειτουργίες που εκτελούνται

- Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (απαιτείται $RegWrite = 1$)

- LW $RF(Rd) = DR$ (το DR έχει έγκυρη τιμή στο Βήμα 5)
- ALU $RF(Rd) = E$ (το E έχει έγκυρη τιμή στο Βήμα 4)
- ALUI $RF(Rd) = E$ (το E έχει έγκυρη τιμή στο Βήμα 4)
- JALR/JR $RF(Rd) = PCplus4$ (το $PCplus4$ παράγεται στο Βήμα 1)
- LUI $RF(Rd) = I$ (το I έχει έγκυρη τιμή στο Βήμα 3)



Δεν απαιτούνται επιπλέον μη αρχιτεκτονικοί καταχωρητές για την εκτέλεση του 5^{ου} βήματος. $PCplus4 = PC+4$

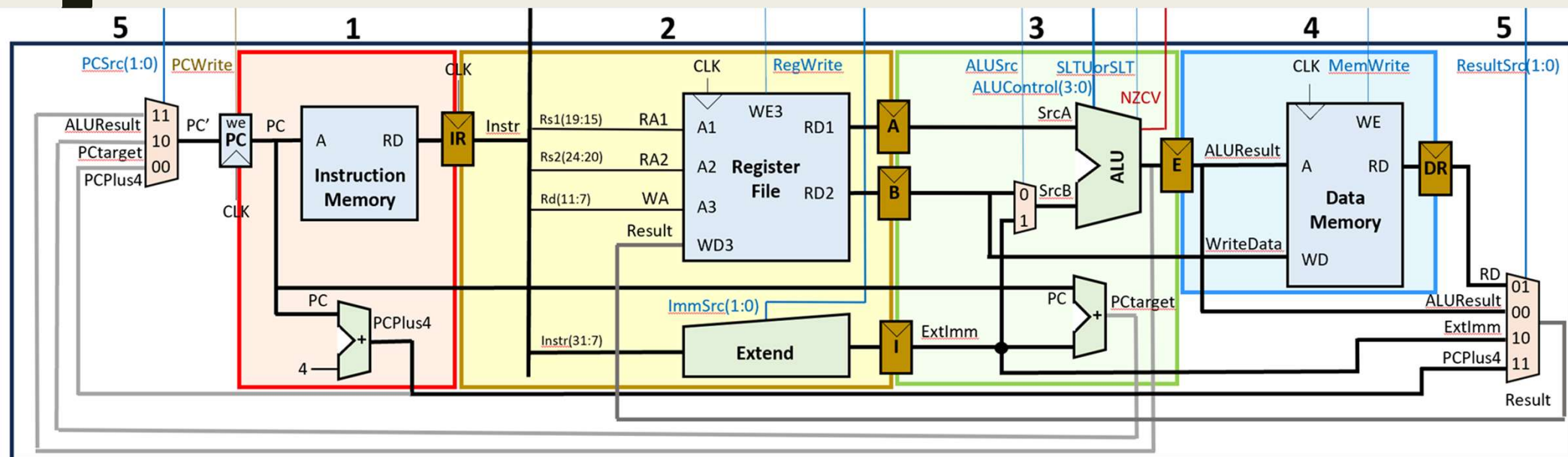
Επεξεργαστής πολλών κύκλων: μελέτη διαδρομής δεδομένων

■ Βήμα 5: Λειτουργίες που εκτελούνται

- Επιλογή διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί (απαιτείται $PCWrite = 1$)

- LW/SW $PC = PC + 4$ (το $PC+4$ παράγεται στο Βήμα 1)
- ALU/ALUI $PC = PC + 4$ (το $PC+4$ παράγεται στο Βήμα 1)
- BEQ/BNE $PC = PC_{target}$ (if $Z=1/0$) (το PC_{target} (BTA) παράγεται στο Βήμα 3)
- BEQ/BNE $PC = PC + 4$ (if $Z=0/1$) (το $PC+4$ παράγεται στο Βήμα 1)
- JALR/JR $PC = ALUResult$ (το $ALUResult$ παράγεται στο Βήμα 3)
- LUI $PC = PC + 4$ (το $PC+4$ παράγεται στο Βήμα 1)

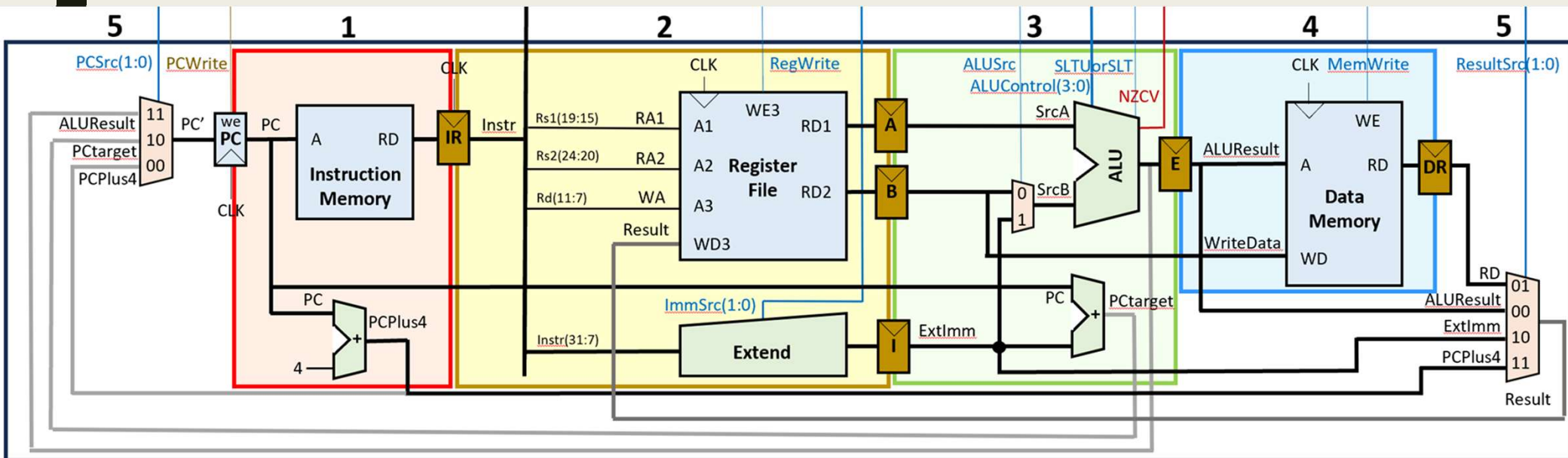
$PC_{target} = BTA = PC + Sign_Ext(instr(31) \& instr(7) \& instr(30:25) \& instr(11:8) \& '0')$



Δεν απαιτούνται επιπλέον καταχωρητές για την εκτέλεση του 5^{ου} βήματος. $ALUResult = A+I$

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση της εντολής LW (τύπου I)

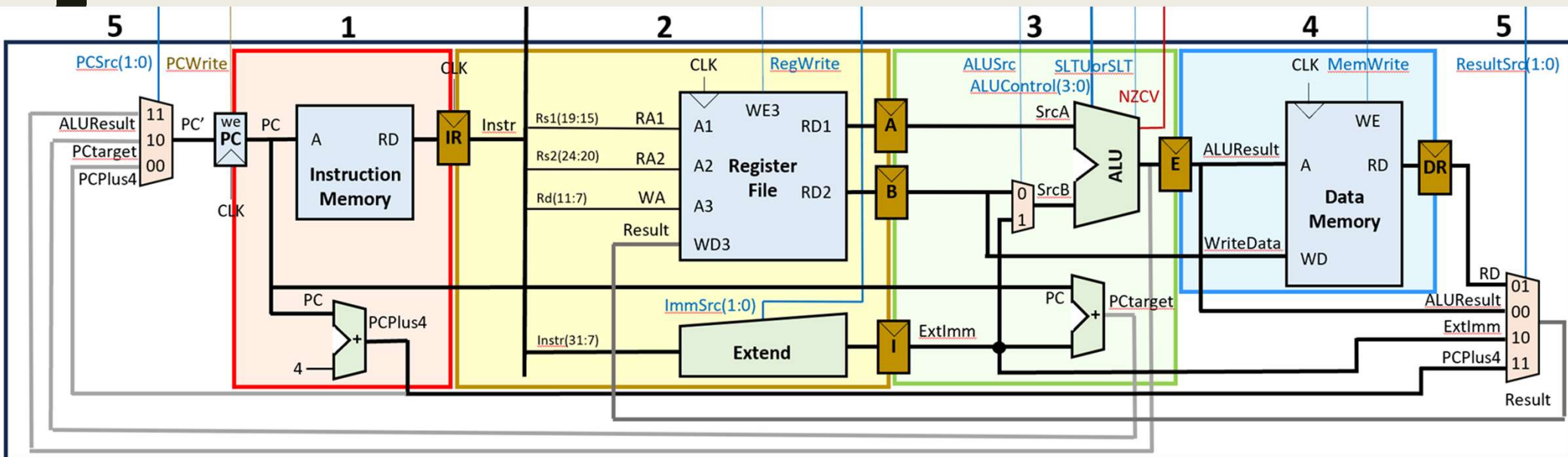
Κύκλος	Βήμα	RTL περιγραφή	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	$IR = IM(PC)$, PCplus4 = $PC + 4$	0	0	0
2	2	$A = RF(Rs1)$, $I = Sign_Ext(instr(31:20))$	0	0	0
3	3	$E = A + I$	0	0	0
4	4	$DR = DM(E)$	0	0	0
5	5	$RF(Rd) = DR$, $PC = \text{PCplus4}$	1	1	0



Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση της εντολής SW (τύπου S)

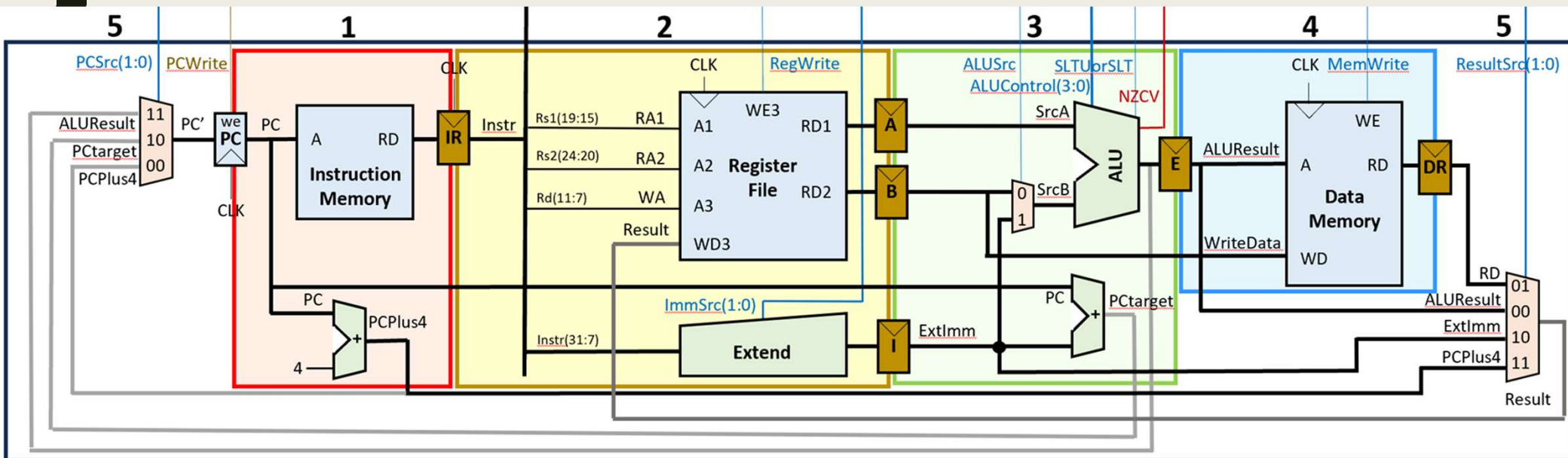
Κύκλος	Βήμα	RTL περιγραφή	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), B = RF(Rs2), I = Sign_Ext(instr(31:25)&instr(11:7))	0	0	0
3	3	E = A + I, WriteData = B	0	0	0
4	4, 5	DM(E) = WriteData , PC = PCplus4	1	0	1



Τα σήματα **PCplus4** και **WriteData** δεν αποθηκεύονται σε μη αρχιτεκτονικούς καταχωρητές

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών ALU (τύπου R)

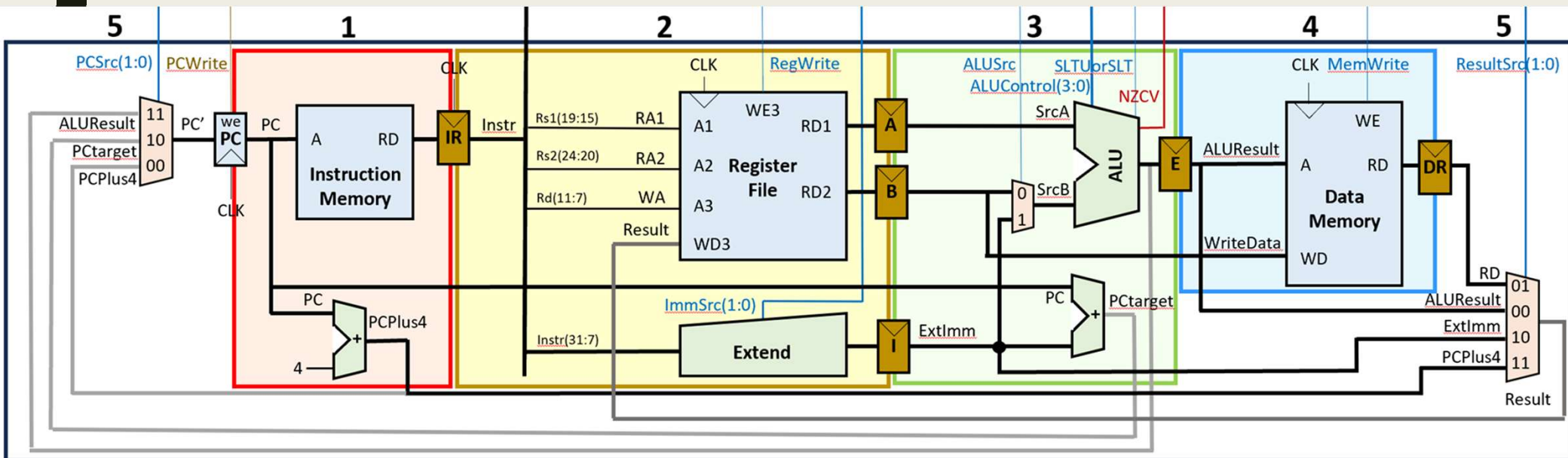
Κύκλος	Βήμα	RTL περιγραφή (ADD, SUB, AND, OR, XOR)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), B = RF(Rs2)	0	0	0
3	3	E = A +/-and/or/xor B	0	0	0
4	5	RF(Rd) = E, PC = PCplus4	1	1	0



Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών ALU (τύπου R)

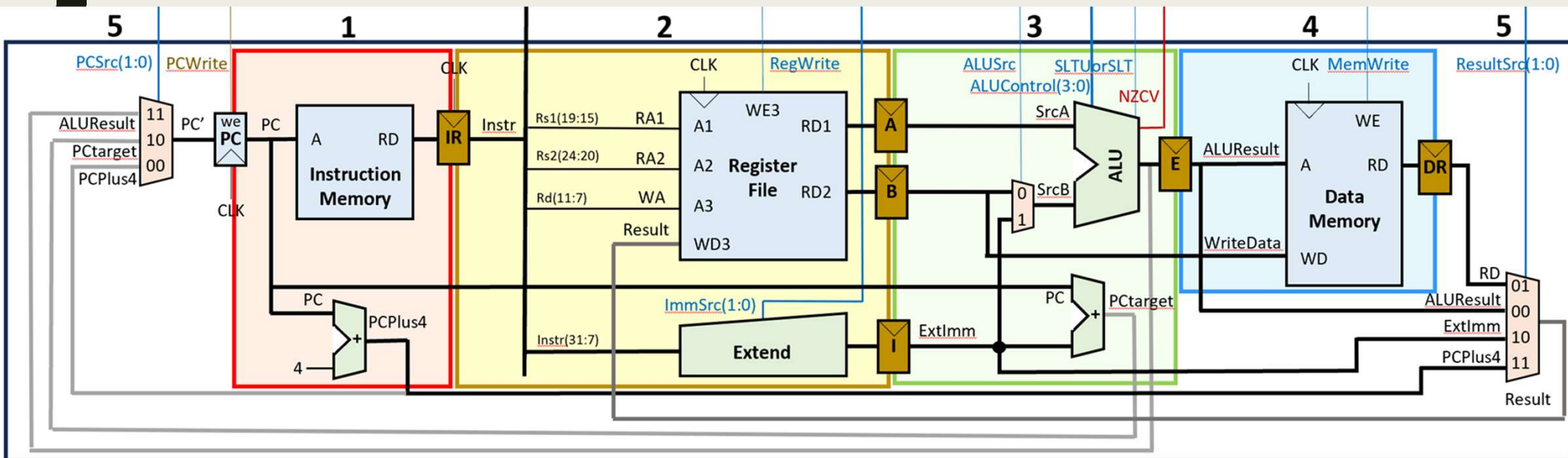
Κύκλος	Βήμα	RTL περιγραφή (SLT/SLTU)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), B = RF(Rs2)	0	0	0
3	3	E = 1, if rs1<rs2 (Signed/Unsigned), else 0	0	0	0
4	5	RF(Rd) = E, PC = PCplus4	1	1	0



Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών ALU (τύπου R)

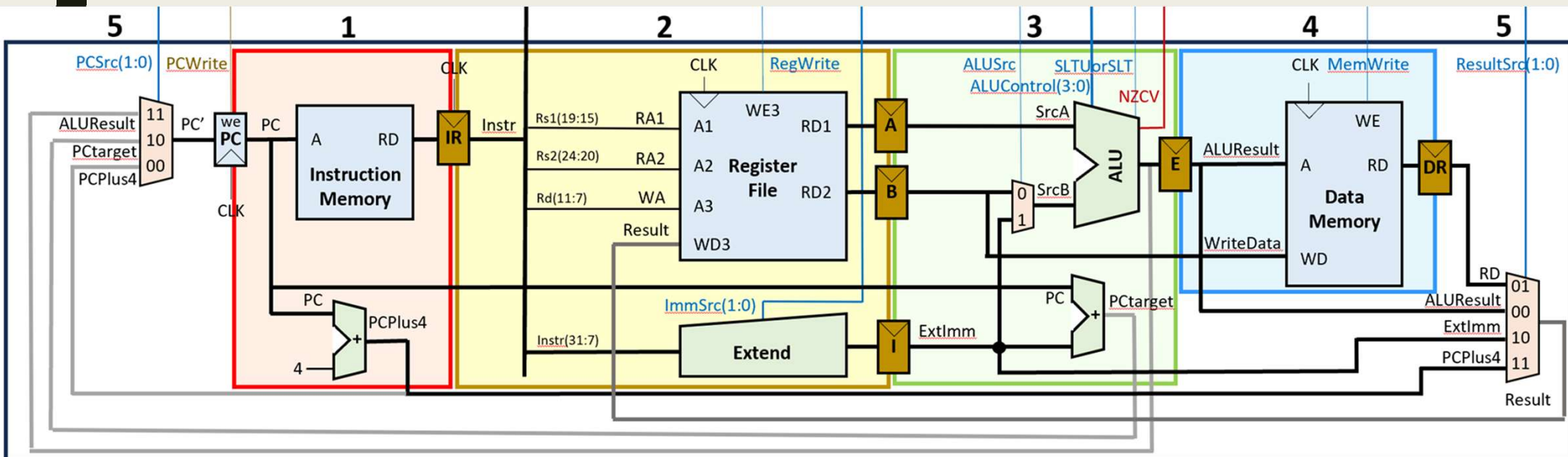
Κύκλος	Βήμα	RTL περιγραφή (SLL, SRL, SRA)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), B = RF(Rs2)	0	0	0
3	3	E = Rs1 <</>>/>>> Rs2(4:0)	0	0	0
4	5	RF(Rd) = E, PC = PCplus4	1	1	0



Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών ALUI (τύπου I)

Κύκλος	Βήμα	RTL περιγραφή (ADDI, ANDI, ORI, XORI)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), I = Sign_Ext(instr(31:20))	0	0	0
3	3	E = A +/and/or/xor I	0	0	0
4	5	RF(Rd) = E, PC = PCplus4	1	1	0

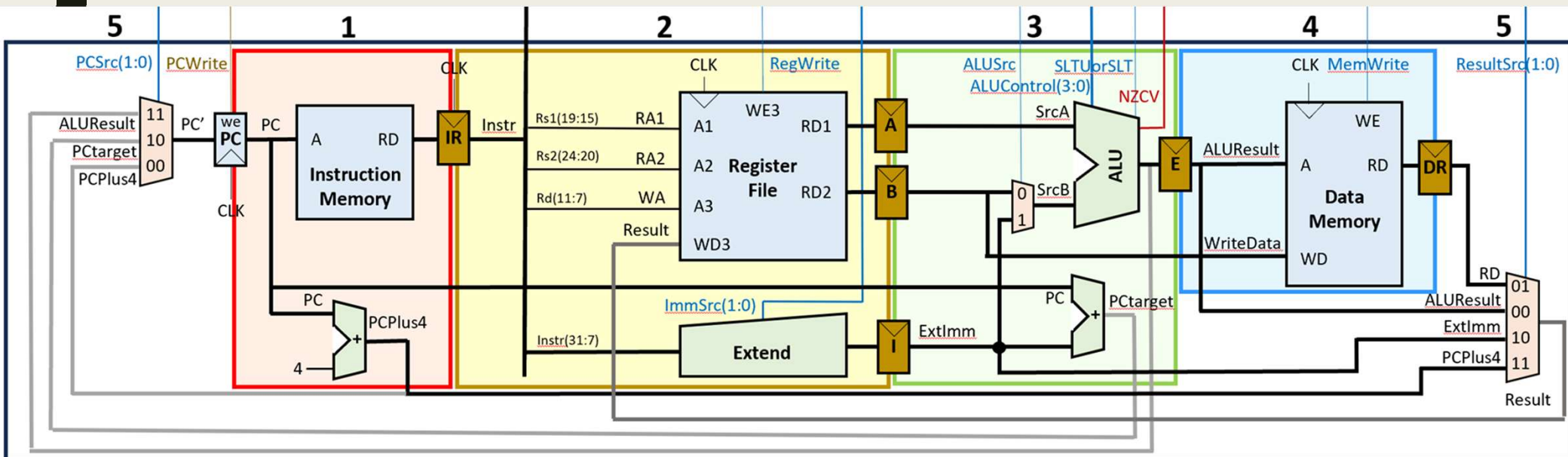


Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών BEQ/BNE (τύπου B)

Κύκλος	Βήμα	RTL περιγραφή (BEQ/BNE)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1), B = RF(Rs2) I = Sign_Ext(instr(31)&instr(7)&instr(30:25)&instr(11:8)&'0')	0	0	0
3	3,5	Z = 1, if A=B, else 0, PCtarget = PC + I (BTA) PC = PCtarget , if Z = 1/0, else PC = PCplus4	1	0	0

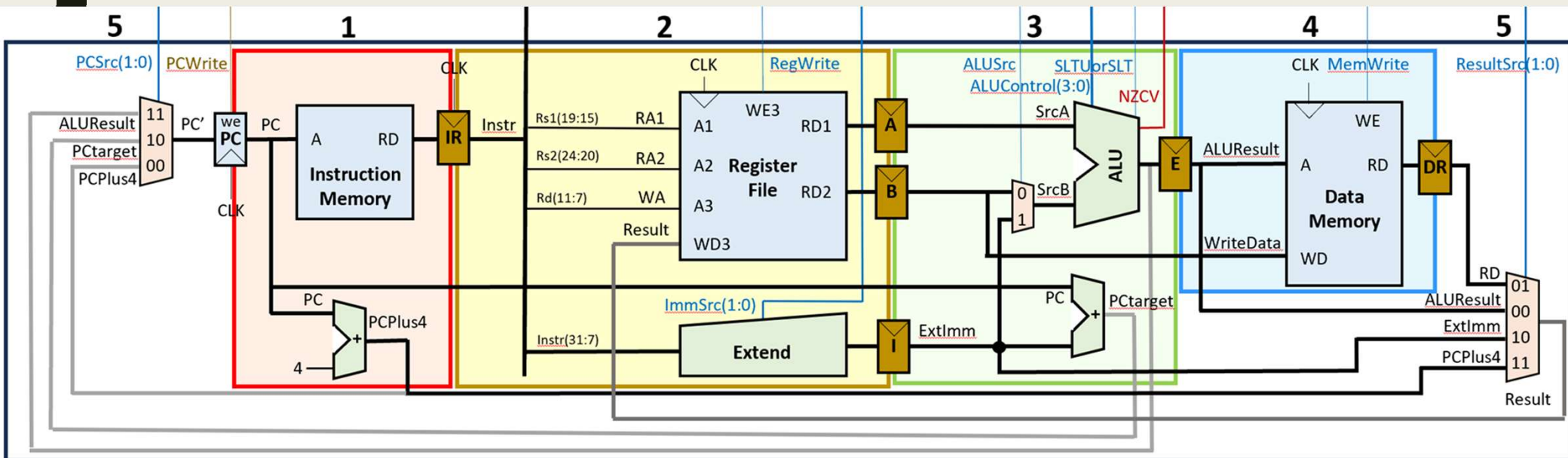
PCtarget = BTA = PC + Sign_Ext(instr(31)&instr(7)&instr(30:25)&instr(11:8)&'0')



Τα σήματα **PCplus4**, **Z (flag)** και **PCtarget** δεν αποθηκεύονται σε μη αρχιτεκτονικούς καταχωρητές

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών JALR/JR (Τύπου I)

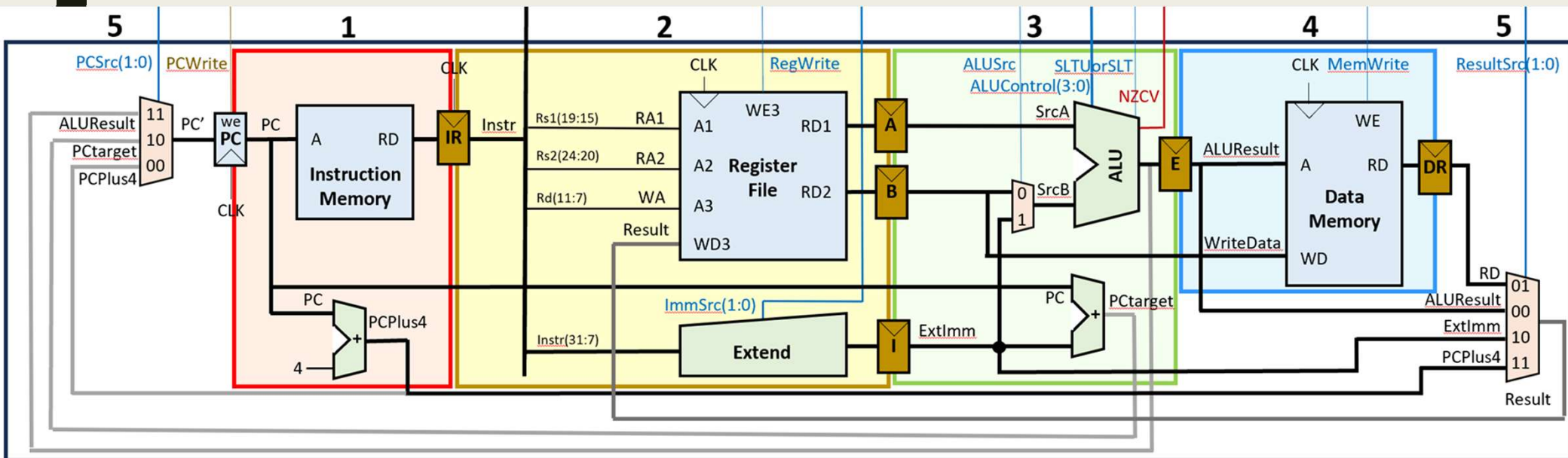
Κύκλος	Βήμα	RTL περιγραφή (JALR/JR)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	A = RF(Rs1) I = Sign_Ext(instr(31:20))	0	0	0
3	3,5	ALUResult = A + I, RF(Rd) = PCplus4 , PC = ALUResult	1	1	0



Τα σήματα **PCplus4** και **ALUResult** δεν αποθηκεύονται σε μη αρχιτεκτονικούς καταχωρητές

Επεξεργαστής πολλών κύκλων: βηματική εκτέλεση των εντολών LUI (Τύπου U)

Κύκλος	Βήμα	RTL περιγραφή (JALR/JR)	Σήματα έγκρισης εγγραφής PCWrite RegWrite MemWrite		
1	1	IR = IM(PC), PCplus4 = PC + 4	0	0	0
2	2	I = instr(31:12)&0x000	0	0	0
3	3,5	RF(Rd) = I PC = PCplus4	1	1	0



Το σήμα **PCplus4** δεν αποθηκεύεται σε μη αρχιτεκτονικό καταχωρητή

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Η **μονάδα ελέγχου** (control unit) παράγει τα κατάλληλα σήματα ελέγχου για τον χρονισμό του επεξεργαστή με τη χρήση:
 - **μηχανής πεπερασμένων καταστάσεων** για τα σήματα **έγκρισης εγγραφής** **PCWrite**, **RegWrite** και **MemWrite**, ώστε να έχουν την σωστή τιμή 0 ή 1 σε κάθε κύκλο εκτέλεσης της εντολής
 - **συνδυαστικής λογικής** για τα υπόλοιπα σήματα ελέγχου που παραμένουν σταθερά σε όλους τους κύκλους εκτέλεσης της εντολής, όπως στην περίπτωση του επεξεργαστή ενός κύκλου
 - μέχρι την ολοκλήρωση της βηματικής εκτέλεσης της εντολής παραμένουν σταθερά
- Είσοδοι της μονάδας ελέγχου (όπως και στον επεξεργαστή ενός κύκλου):
 - πεδίο **op** της εντολής (*Instr(6:0)*) για τον προσδιορισμό του τύπου της εντολής και του είδους της πράξης
 - πεδίο **funct3** της εντολής (*Instr(14:12)*) για τη δήλωση της λειτουργίας/πράξης
 - πεδίο **funct7(5)** της εντολής (*Instr(30)*) ως επέκταση του πεδίου **funct3**
- Η μονάδα ελέγχου ελέγχει εάν ικανοποιείται η συνθήκη για τις εντολές Branch και παράγει το κατάλληλο σήμα ελέγχου **PCSrc**
 - Για την υλοποίηση των εντολών *BEQ/BNQ* ως είσοδος της μονάδας ελέγχου χρησιμοποιείται μόνο το *ALU flag* **Z** (zero)

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

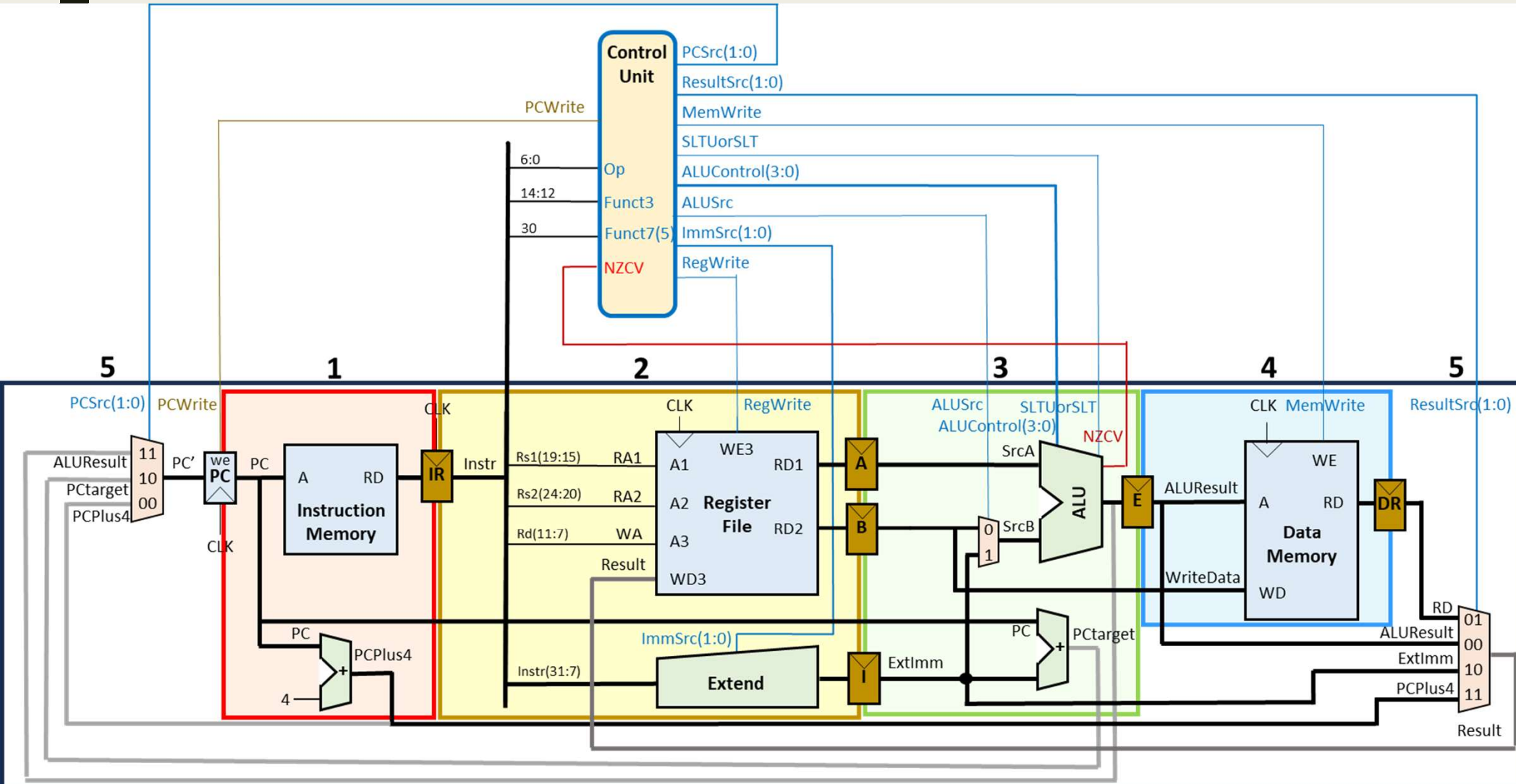
- Έξοδοι της μονάδας ελέγχου που παραμένουν σταθερές κατά την εκτέλεση όλων των βημάτων της εντολής:
 - σήματα επιλογής πολυπλεκτών
 - **ALUSrc** (mux ALU SrcB)
 - **ResultSrc(1:0)** (mux Result) (για ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών)
 - σήματα ελέγχου λογικής
 - **ALUControl(3:0)** (επιλογή πράξης στη μονάδα ALU)
 - **SLTUorSLT** (διάκριση μεταξύ SLT και SLTU στη μονάδα ALU)
 - **ImmSrc(1:0)** (επιλογή επέκτασης προσήμου στη μονάδα Extend)
 - σήμα επιλογής διεύθυνσης επόμενης εντολής
 - **PCSrc(1:0)** (mux PC') – ελέγχεται από το ALU flag **Z** για BEQ/BNE
- Παράγονται με τα ίδια συνδυαστικά κυκλώματα που μελετήσαμε στον επεξεργαστή ενός κύκλου

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Έξοδοι της μονάδας ελέγχου που ενεργοποιούνται (παίρνουν την τιμή 1) σε συγκεκριμένο Βήμα (κύκλο) κατά την εκτέλεση της εντολής:
 - τα σήματα έγκρισης εγγραφής και το βήμα στο οποίο ενεργοποιούνται
 - **PCWrite** (μετρητής προγράμματος PC – Βήμα 5)
 - **RegWrite** (αρχείο καταχωρητών RF – Βήμα 5)
 - **MemWrite** (μνήμη δεδομένων DM – Βήμα 4)
- Παράγονται από μηχανή πεπερασμένων καταστάσεων (FSM)

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Ολοκληρωμένος επεξεργαστής πολλών κύκλων



Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Στη συνέχεια θα σχεδιάσουμε τη μονάδα ελέγχου που απαρτίζεται από τις ακόλουθες υπομονάδες:

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής (*Instr*(6:2) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc(1:0)**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc(1:0)** και το εσωτερικό σήμα **rop(2:0)**

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και τα πεδία **funct3** (*Instr*(14:12) και **funct7(5)** (*Instr*(30) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl(3:0)** και **SLTUorSLT**

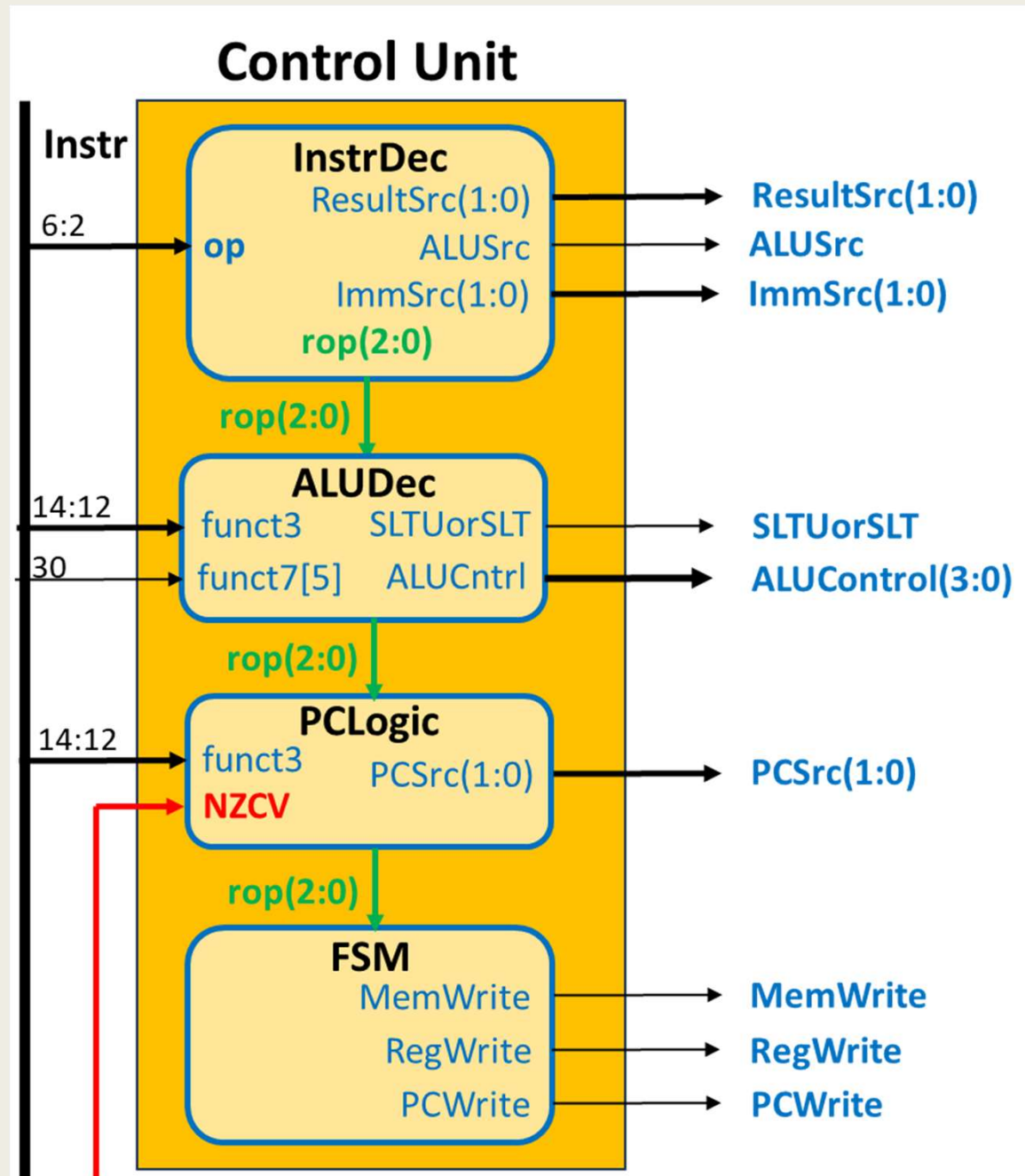
3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και το πεδίο **funct3** (*Instr*(14:12), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές *Branch* με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των *BEQ/BNE*), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc(1:0)**

4. Μηχανή πεπερασμένων καταστάσεων (FSM)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)** και παράγει τα κατάλληλα σήματα έγκρισης εγγραφής **PCWrite**, **RegWrite** και **MemWrite**

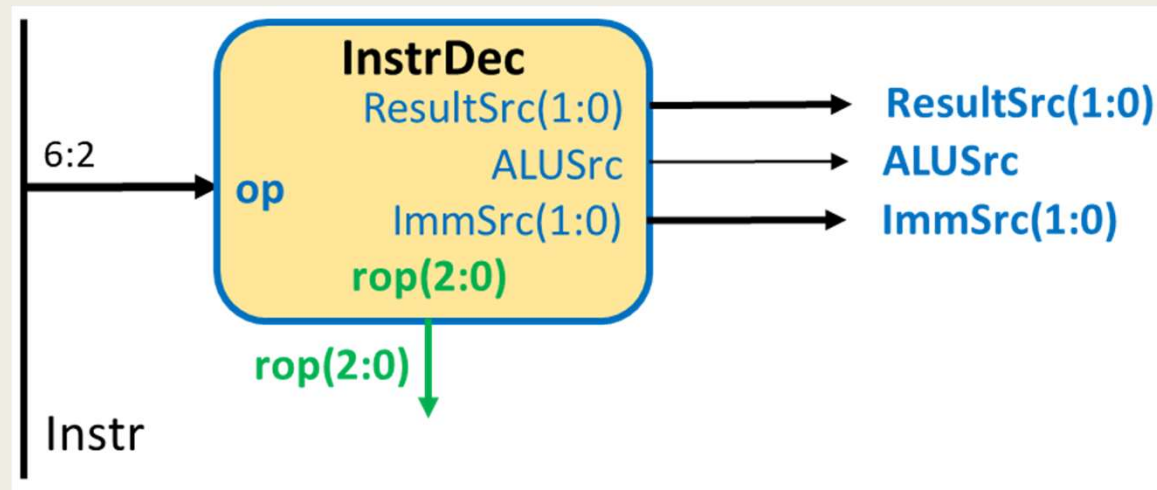
Επεξεργαστής πολλών κύκλων: ολοκληρωμένη μονάδα ελέγχου RISC-V πολλών κύκλων



Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

- η υπομονάδα που αποκωδικοποιεί το πεδίο **op** της εντολής (*Instr*(6:2) και παράγει τα σήματα επιλογής πολυπλεκτών **ALUSrc** και **ResultSrc**, το σήμα επιλογής επέκτασης προσήμου **ImmSrc(1:0)** και το εσωτερικό σήμα **rop(2:0)**



Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

1. Κύριος αποκωδικοποιητής εντολής (InstrDec)

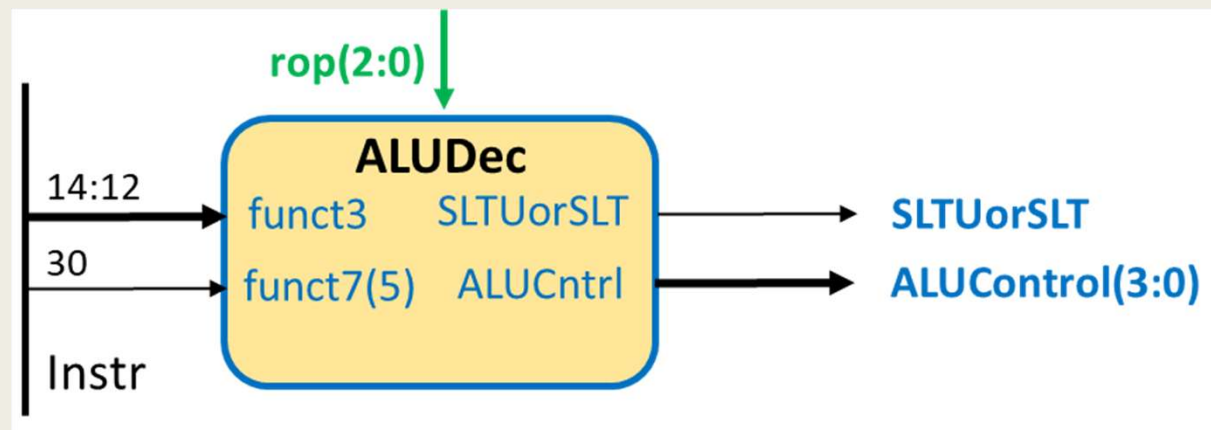
Εντολή	op(6:2)	rop(2:0)	Result Src(1:0)	ALU Src	Imm Src(1:0)
LW	00000	000	01	1	00
ALUI	00100	001	00	1	00
SW	01000	010	00	1	01
ALU	01100	011	00	0	XX
Branch	11000	100	00	0	10
JALR	11001	101	11	1	00
LUI	01101	11X	10	X	11

Τα σήματα **MemWrite** και **RegWrite** παράγονται πλέον στον FSM.
Τα υπόλοιπα σήματα παραμένουν ίδια με τον επεξεργαστή του ενός κύκλου

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και τα πεδία **funct3** (*Instr*(14:12) και **funct7(5)** (*Instr*(30) της εντολής, και παράγει τα σήματα ελέγχου της μονάδας ALU: **ALUControl(3:0)** και **SLTUorSLT**



Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

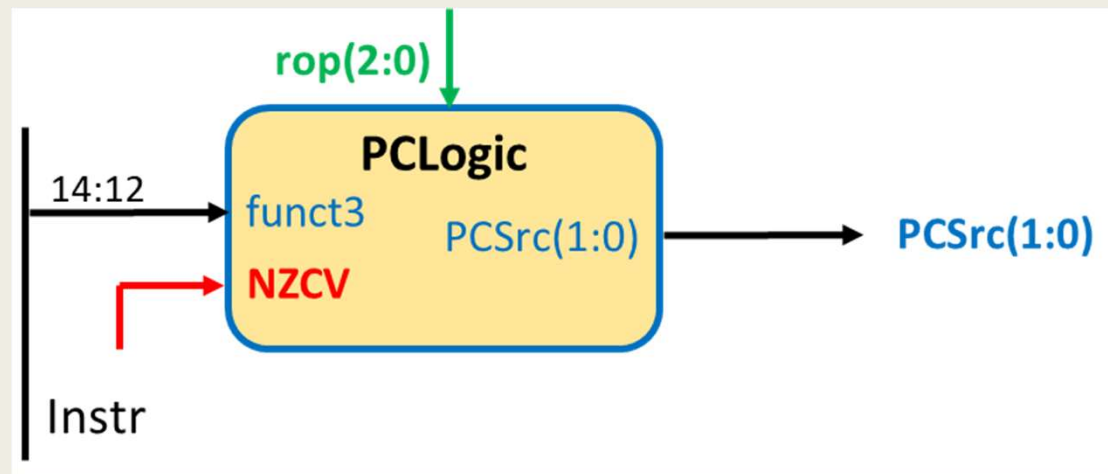
Εντολή	op(6:2)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (3:0)	SLTUorSLT
ADD	01100	011	000	0	add	0000	X
SUB	01100	011	000	1	sub	0001	X
AND	01100	011	111	0	and	0100	X
OR	01100	011	110	0	or	0101	X
XOR	01100	011	100	0	xor	011X	X
SLL	01100	011	001	0	sll	1X0X	X
SRL	01100	011	101	0	srl	1X10	X
SRA	01100	011	101	1	sra	1X11	X
SLT	01100	011	010	0	slt	001X	0
SLTU	01100	011	011	0	sltu	001X	1

Παραμένει ίδιος με τον επεξεργαστή ενός κύκλου

Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)**, καθώς και το πεδίο **funct3** (*Instr*(14:12), εξετάζει εάν ικανοποιείται ή όχι η συνθήκη στις εντολές *Branch* με βάση τις τιμές των **ALU flags** (μόνο του **Z** στην περίπτωση των *BEQ/BNE*), και παράγει το κατάλληλο σήμα ελέγχου **PCSrc(1:0)**



Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

Εντολή	op(6:2)	rop(2:0)	funct3	NZCV	PCSrc(1:0)
LW	00000	000	XXX	XXXX	00
ALUI	00100	001	XXX	XXXX	00
SW	01000	010	XXX	XXXX	00
ALU	01100	011	XXX	XXXX	00
BEQ	11000	100	000	X0XX	00
BEQ	11000	100	000	X1XX	10
BNE	11000	100	001	X0XX	10
BNE	11000	100	001	X1XX	00
JALR	11001	101	XXX	XXXX	11
LUI	01101	11X	XXX	XXXX	00

Παραμένει ίδια με τον επεξεργαστή ενός κύκλου

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

4. Μηχανή πεπερασμένων καταστάσεων (FSM)

- η υπομονάδα λαμβάνει το εσωτερικό σήμα **rop(2:0)** και παράγει τα κατάλληλα σήματα έγκρισης εγγραφής **PCWrite**, **RegWrite** και **MemWrite**
 - Σύμφωνα με το σήμα **rop(2:0)** υλοποιούνται 7 διαφορετικές ομάδες εντολών
 - Οι εντολές της ίδιας ομάδας έχουν τις ίδιες απαιτήσεις σε κύκλους (βήματα) εκτέλεσης εντολής και τιμές των σημάτων έγκρισης εγγραφής

Εντολή	op(6:2)	rop(2:0)
LW	00000	000
ALUI	00100	001
SW	01000	010
ALU	01100	011
Branch	11000	100
JALR	11001	101
LUI	01101	11X

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Βήματα (κύκλοι) εκτέλεσης των εντολών ανά ομάδα εντολών
 - Κάθε εντολή εκτελείται στα ακόλουθα πέντε βήματα:
 1. FETCH: Προσκόμιση εντολής και υπολογισμός διεύθυνσης της επόμενης εντολής (PC+4)
 2. DECODE: Αποκωδικοποίηση εντολής και ανάγνωση αρχείου καταχωρητών
 3. EXECUTE: Εκτέλεση πράξεων στη μονάδα ALU
 4. MEMORY: Ανάγνωση ή εγγραφή στη μνήμη δεδομένων
 5. WRITEBACK: Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών και επιλογή της διεύθυνσης της επόμενης εντολής
 - κατά την εκτέλεση μίας εντολής πιθανώς να μην απαιτούνται όλα τα βήματα ή να εκτελούνται παράλληλα στον ίδιο κύκλο, όσα δεν δημιουργούν εξαρτήσεις

Κύκλοι	LW	SW	ALU	ALUI	Branch	JALR/JR	LUI
1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2
3	3	3	3	3	3, 5	3,5	3,5
4	4	4, 5	5	5			
5	5						

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Τιμές των σημάτων έγκρισης εγγραφής ανά κύκλο και ανά ομάδα εντολών
 - Απαιτούνται τα ακόλουθα 3 σήματα έγκρισης εγγραφής
 - **PCWrite** (μετρητής προγράμματος PC – ενεργοποιείται στο Βήμα 5)
 - **RegWrite** (αρχείο καταχωρητών RF – ενεργοποιείται στο Βήμα 5)
 - **MemWrite** (μνήμη δεδομένων DM – ενεργοποιείται στο Βήμα 4)

Κύκλοι	LW PC Reg Mem	SW PC Reg Mem	ALU PC Reg Mem	ALUI PC Reg Mem	Branch PC Reg Mem	JALR/JR PC Reg Mem	LUI PC Reg Mem
1	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
2	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0
3	0 0 0	0 0 0	0 0 0	0 0 0	1 0 0	1 1 0	1 1 0
4	0 0 0	1 0 1	1 1 0	1 1 0			
5	1 1 0						

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

4. Μηχανή πεπερασμένων καταστάσεων (FSM)

- Δημιουργία διαγράμματος μεταβολής κατάστασης
 - Προσδιορισμός διακριτών καταστάσεων
 - Δύο τρέχουσες καταστάσεις χαρακτηρίζονται ως διακριτές μεταξύ τους, εάν :
 - τουλάχιστον μία έξοδος, που εξαρτάται αποκλειστικά από την τρέχουσα κατάσταση, έχει διαφορετική τιμή, ή/και
 - έχουν τις ίδιες τιμές στις εξόδους, αλλά διαφορετική επόμενη κατάσταση που είναι ανεξάρτητη από τις εισόδους

Κύκλοι	LW PC Reg Mem	SW PC Reg Mem	ALU PC Reg Mem	ALUI PC Reg Mem	Branch PC Reg Mem	JALR/JR PC Reg Mem	LUI PC Reg Mem
1	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0
2	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0
3	S2 / 0 0 0	S2 / 0 0 0	S2 / 0 0 0	S2 / 0 0 0	S4c / 1 0 0	S4a / 1 1 0	S4a / 1 1 0
4	S3 / 0 0 0	S4b / 1 0 1	S4a / 1 1 0	S4a / 1 1 0			
5	S4a / 1 1 0						

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

4. Μηχανή πεπερασμένων καταστάσεων (FSM)

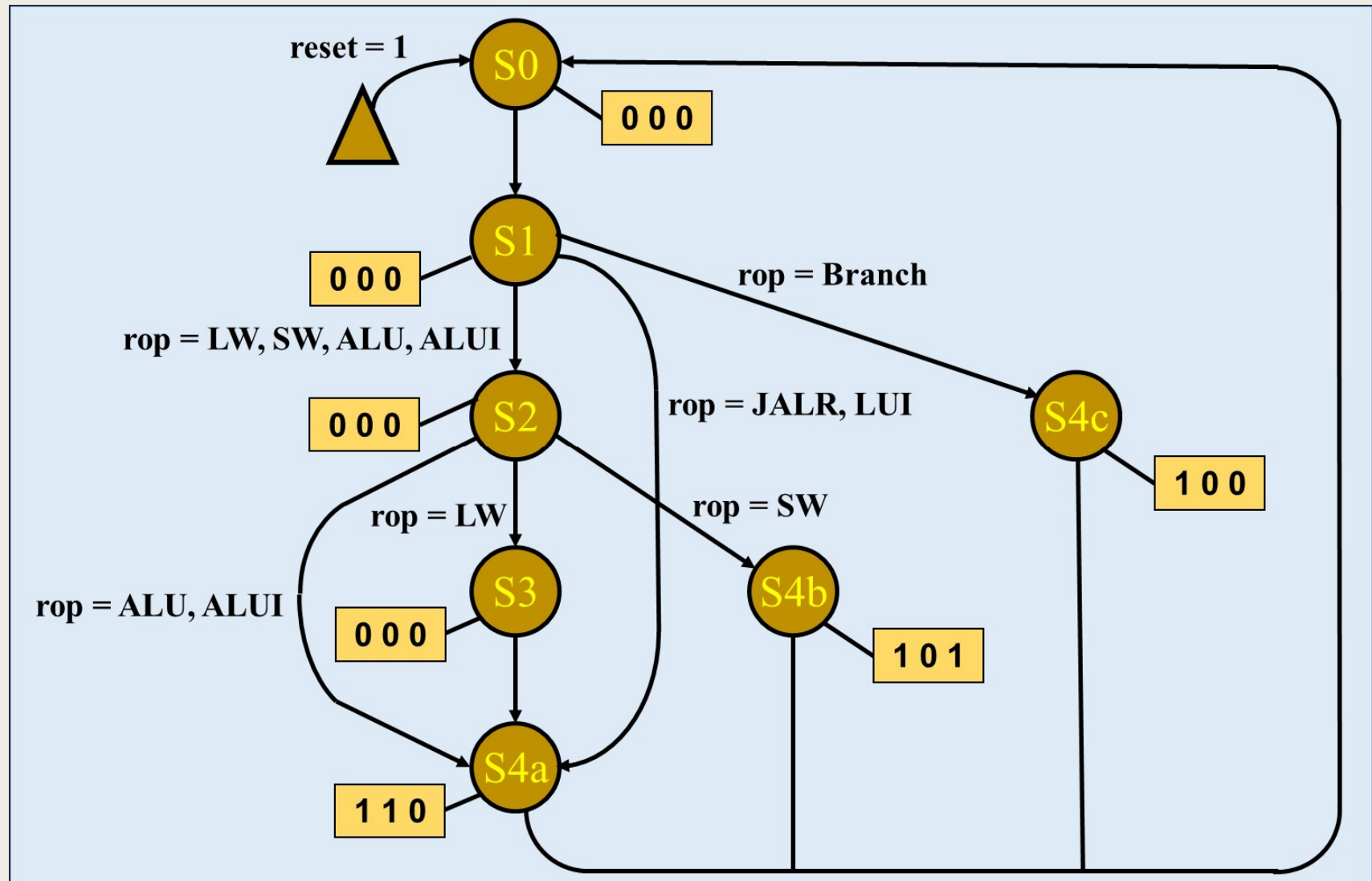
- Δημιουργία διαγράμματος μεταβολής κατάστασης
 - Περιγραφή διακριτών καταστάσεων (10 καταστάσεις)
 - **S0**: τρέχουσα κατάσταση στο βήμα *FETCH* (η αρχική κατάσταση με *RESET*)
 - **S1**: τρέχουσα κατάσταση στο βήμα *DECODE*
 - **S2**: τρέχουσα κατάσταση στο βήμα *EXECUTE*
 - **S3**: τρέχουσα κατάσταση στο βήμα *MEMORY*
 - **S4a/S4b/S4c**: τρέχουσα κατάσταση με την οποία ολοκληρώνεται η εκτέλεση της εντολής, διαφοροποιούνται στις τιμές των σημάτων έγκρισης εγγραφής. Επιστρέφει στην αρχική κατάσταση *S0*

Κύκλοι	LW PC Reg Mem	SW PC Reg Mem	ALU PC Reg Mem	ALUI PC Reg Mem	Branch PC Reg Mem	JALR/JR PC Reg Mem	LUI PC Reg Mem
1	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0	S0 / 0 0 0
2	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0	S1 / 0 0 0
3	S2 / 0 0 0	S2 / 0 0 0	S2 / 0 0 0	S2 / 0 0 0	S4c / 1 0 0	S4a / 1 1 0	S4a / 1 1 0
4	S3 / 0 0 0	S4b / 1 0 1	S4a / 1 1 0	S4a / 1 1 0			
5	S4a / 1 1 0						

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

4. Μηχανή πεπερασμένων καταστάσεων (FSM)

- Διάγραμμα μεταβολής κατάστασης για FSM τύπου Moore



Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

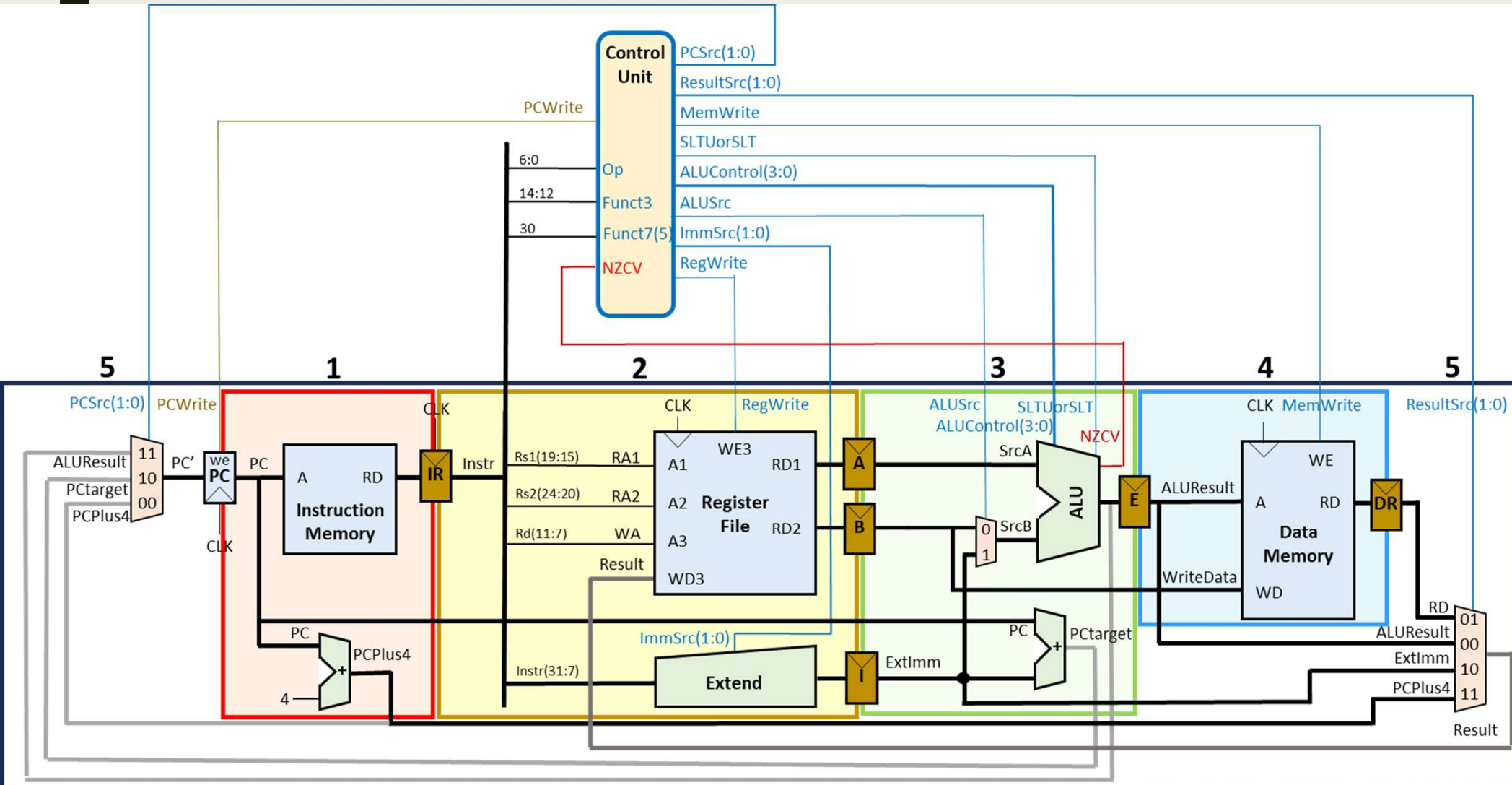
4. Μηχανή πεπερασμένων καταστάσεων (FSM)

- Μονάδα εξόδου για FSM τύπου Moore

Τρέχ. Κατ.	Κώδικας	PC Reg Mem
S0	000	000
S1	001	000
S2	010	000
S3	011	000
S4a	110	110
S4b	101	101
S4c	100	100

Επεξεργαστής πολλών κύκλων: μελέτη μονάδας ελέγχου RISC-V πολλών κύκλων

- Ολοκληρωμένος επεξεργαστής πολλών κύκλων



Ανάλυση επιδόσεων επεξεργαστών

■ Πώς επηρεάζεται η επίδοση

- $\text{CPU}_{\text{time}} = (\# \text{ instructions/program}) \times \text{CPI} \times T_c = \text{seconds/program}$

■ Μέσος αριθμός κύκλων ανά εντολή (cycles per instruction - CPI):

- μέσος αριθμός κύκλων ανά εντολή του ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ενός προγράμματος
- εξαρτάται από το μίγμα των εντολών σε ένα πρόγραμμα (πόσες εντολές είναι επεξεργασίας δεδομένων, μνήμης ή διακλάδωσης)

■ Εντολές ανά κύκλο (instructions per cycle - IPC):

- διεκπεραιωτική ικανότητα - το αντίστροφο του CPI

■ Περίοδος T_c του ρολογιού (clock period):

- το πλήθος των νανοδευτερολέπτων ανά κύκλο ρολογιού
- καθορίζεται από την **κρίσιμη διαδρομή (critical path)** μέσω της λογικής της εκάστοτε μικροαρχιτεκτονικής του επεξεργαστή
- ο τρόπος σχεδίασης της διαδρομής δεδομένων και των μονάδων που την απαρτίζουν επηρεάζουν την **κρίσιμη διαδρομή (critical path)** και συνεπώς την περίοδο του ρολογιού

Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

- Η τιμή του **CPI** στον επεξεργαστή πολλών κύκλων εξαρτάται από:
 - την ποσόστωση (%) των πιθανών κατηγοριών εντολών που εκτελούνται σε ένα συγκεκριμένο πρόγραμμα
 - από τον αριθμό των κύκλων που απαιτούνται κατά την εκτέλεση των εντολών ανά κατηγορία εντολών
- Παράδειγμα υπολογισμού CPI:

Κατηγορία εντολών	Αριθμός κύκλων	Ενδεικτική ποσόστωση	Υπολογισμός CPI
ALU/ALUI	4	50%	2,00
LW	5	20%	1,00
SW	4	10%	0,40
Branch	3	6%	0,18
JALR/JR	3	4%	0,12
LUI	3	10%	0,30
		100%	4,00

Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

- Η περίοδος του CLK καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του **πιο αργού βήματος της εντολής** μέσω της λογικής της μικροαρχιτεκτονικής πολλών κύκλων
 - Εξετάζουμε τρεις περιπτώσεις κρίσιμων διαδρομών
 - **Περίπτωση 1:** Η εκτέλεση του 3^{ου} κύκλου των εντολών Branch
 - $Z = 1$, if $A=B$, else 0, (υπολογισμός Z (flag) στην μονάδα ALU)
 - $PC = PC+I$ (BTA), if $Z = 1/0$, else $PC = PC+4$
 - **Περίπτωση 2:** Η ανάγνωση του Αρχείου Καταχωρητών (RF)
 - $A = RF(Rs1)$,
 - $B = RF(Rs2)$
 - **Περίπτωση 3:** Η ανάγνωση της Μνήμης Δεδομένων (MD) – ομοίως για ID
 - $DR = DM(E)$

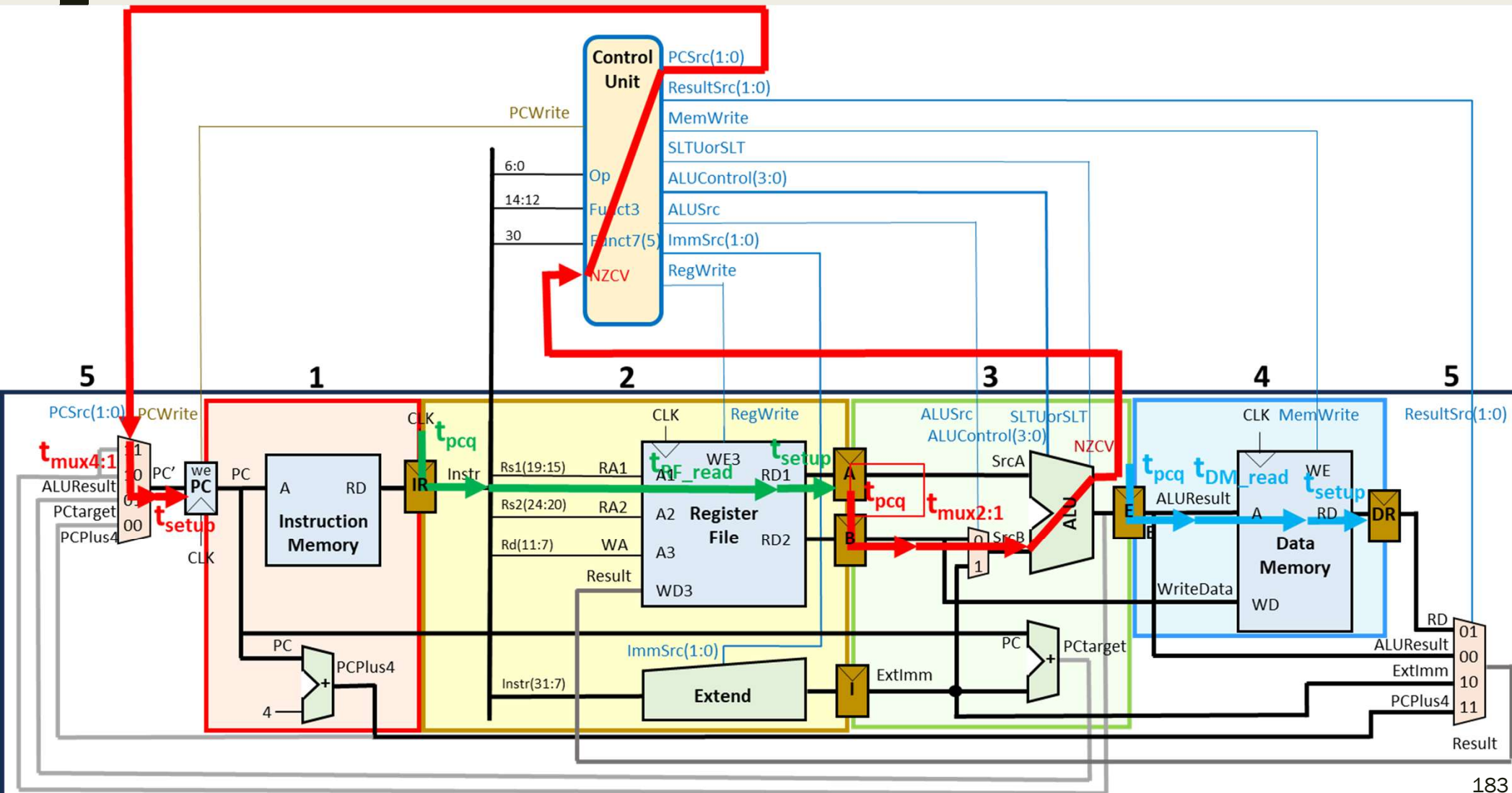
Ενδεικτικές τιμές σε τεχνολογία CMOS 7-nm.

Στοιχεία	Παράμετρος	Καθυστέρηση (ps)
Καθυστέρηση από το CLK έως την έξοδο Q (καταχωρητής)	T_{pcq}	40
Χρόνος σταθεροποίησης καταχωρητή	T_{setup}	50
Χρόνος σταθεροποίησης αρχείου καταχωρητών	t_{RF_setup}	60
Καθ. διάδ. στην μονάδα ελέγχου InstrDec/ALUDec/PCLogic	$t_{IND}/t_{ALUD}/t_{PClogic}$	25/35/35(10 από NZCV)
Καθυστέρηση διάδοσης (control path) στη μονάδα Extend	t_{Ext}	35
Καθυστέρηση διάδοσης πολυπλέκτη 2σε1/4σε1/8σε1	$T_{mux2:1/4:1/8:1}$	20/30/50
Καθυστέρηση διάδοσης στη μονάδα ALU/στον αθροιστή	t_{ALU}/t_{adder}	150/100
Χρόνος ανάγνωσης από τη μνήμη εντολών (ROM)	t_{IM_read}	200
Χρόνος ανάγνωσης από τη μνήμη δεδομένων (RAM)	t_{DM_read}	200
Χρόνος ανάγνωσης από το αρχείο καταχωρητών (RAM)	t_{RF_read}	100

Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

■ Περίπτωση 1: Η εκτέλεση του 3^{ου} κύκλου των εντολών Branch

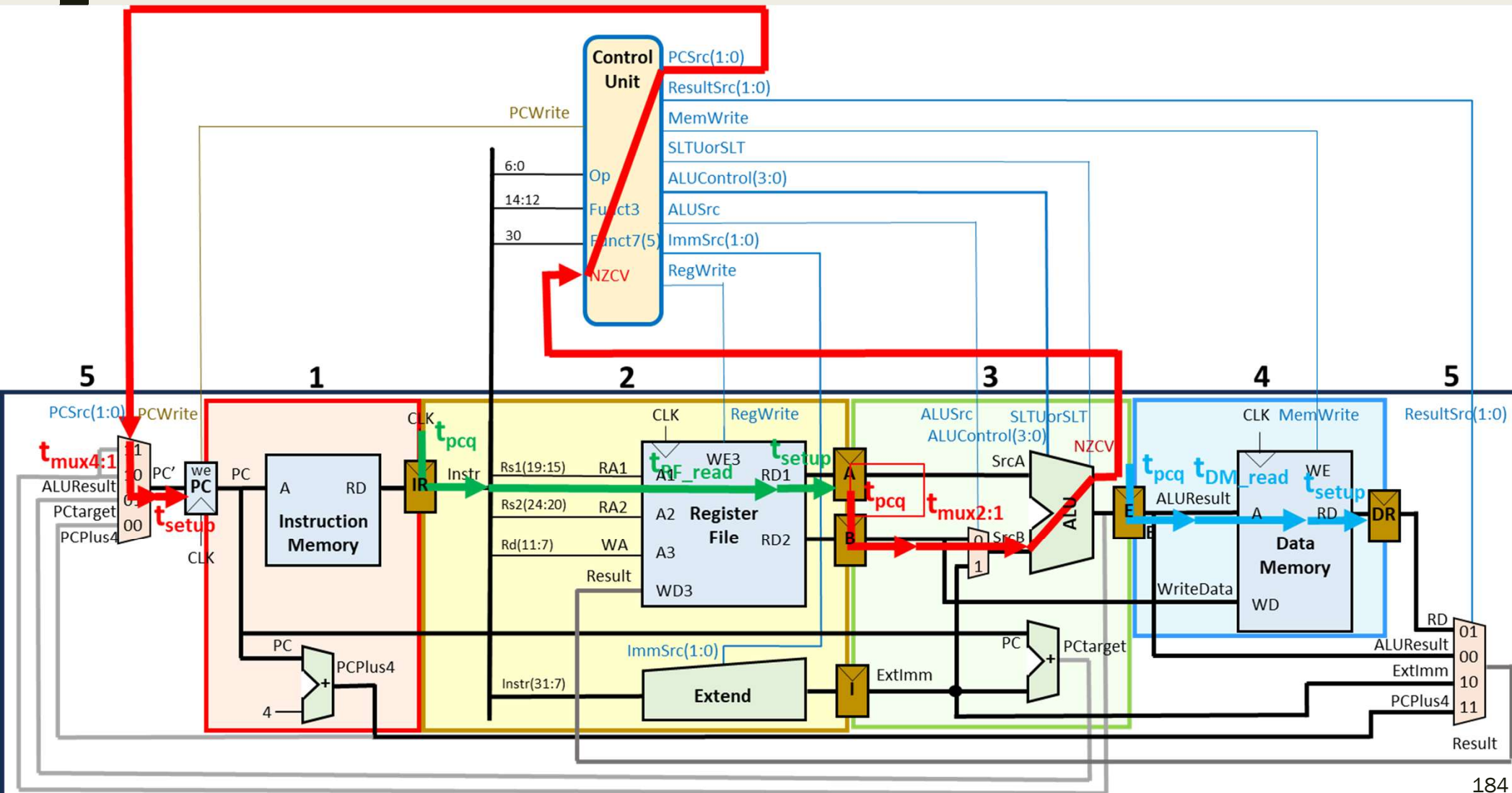
- $T_c = t_{pcq} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup}$
- $T_c = 40 \text{ ps} + 20 \text{ ps} + 150 \text{ ps} + 10 \text{ ps} + 30 \text{ ps} + 50 \text{ ps} = 300 \text{ ps}$



Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

■ Περίπτωση 2: Η ανάγνωση του Αρχείου Καταχωρητών (RF)

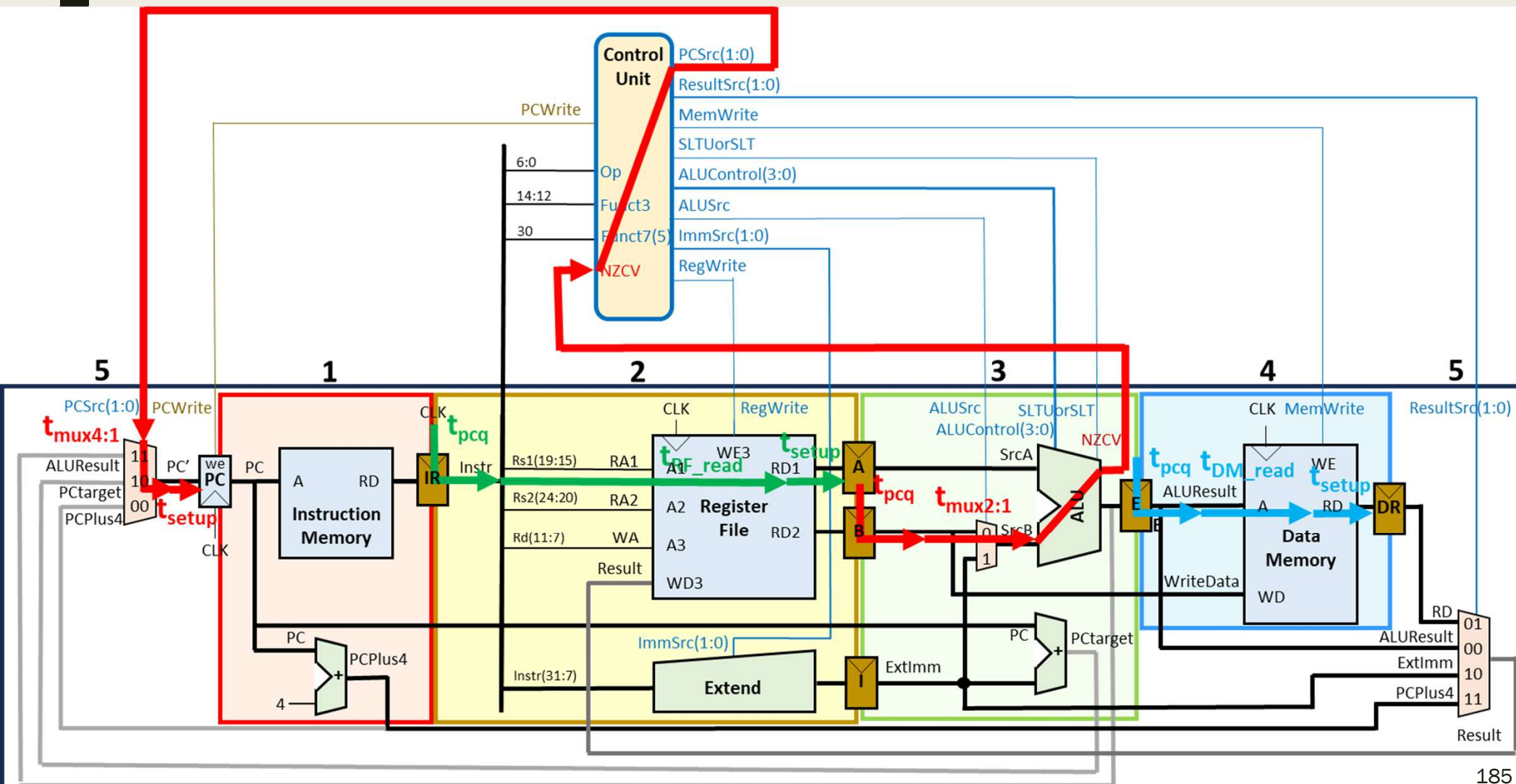
- $T_c = t_{pcq} + t_{RF_read} + t_{setup}$
- $T_c = 40 \text{ ps} + 100 \text{ ps} + 50 \text{ ps} = 190 \text{ ps}$



Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

■ Περίπτωση 3: Η ανάγνωση της Μνήμης Δεδομένων (MD)

- $T_c = t_{pcq} + t_{DM_read} + t_{setup}$
- $T_c = 40 \text{ ps} + 200 \text{ ps} + 50 \text{ ps} = 290 \text{ ps}$



Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

■ Επεξεργαστής ενός κύκλου (CPI = 1)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση της **εντολής LW**
 - $T_c = t_{pcq} + t_{IM_read} + t_{RF_read} + t_{ALU} + t_{DM_read} + t_{mux4:1} + t_{RF_setup}$
- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:
 - $T_c = (40 + 200 + 100 + 150 + 200 + 30 + 60) \text{ ps} = 780 \text{ ps}$
- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:
 - $\text{CPU}_{\text{time}} = (\# \text{ instr/program}) \times \text{CPI} \times T_c = 100 \times 10^9 \times 1 \times 780 \times 10^{-12} = 78 \text{ s}$

■ Επεξεργαστής πολλών κύκλων (CPI =4)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του 3^{ου} κύκλου των εντολών Branch
 - $T_c = t_{pcq} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup}$
- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:
 - $T_c = (40 + 20 + 150 + 10 + 30 + 50) \text{ ps} = 300 \text{ ps}$
- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:
 - $\text{CPU}_{\text{time}} = (\# \text{ instr/program}) \times \text{CPI} \times T_c = 100 \times 10^9 \times 4 \times 300 \times 10^{-12} = 120 \text{ s}$

Ο επεξεργαστής πολλών κύκλων είναι τελικά πιο αργός από τον επεξεργαστή ενός κύκλου

Επεξεργαστής πολλών κύκλων: ανάλυση επιδόσεων

- Ο επεξεργαστής πολλών κύκλων είναι **πιο αργός** από τον επεξεργαστή ενός κύκλου
 - Αν και η πιο αργή εντολή (LW) χωρίστηκε σε πέντε βήματα, ο χρόνος κύκλου του επεξεργαστή πολλών κύκλων δεν βελτιώθηκε ούτε κατά διάνοια στο πενταπλάσιο (είναι κάτω από το στο τριπλάσιο)
 - Δεν έχουν όλα τα βήματα ακριβώς την ίδια καθυστέρηση διάδοσης
 - Τα 90 ps της επιβάρυνσης **δημιουργίας ακολουθίας** (sequencing overhead) προκαλούνται **σε κάθε ξεχωριστό βήμα** εκτέλεσης της εντολής, και **όχι μόνο μία φορά** για την εκτέλεση ολόκληρης της εντολής
 - Ο επεξεργαστής ενός κύκλου έχει $CPI = 1$, ενώ ο επεξεργαστής πολλών κύκλων έχει περίπου τετραπλάσιο CPI ($CPI = 4$)
- Ένας επεξεργαστής πολλών κύκλων γίνεται **γρήγορος** μόνο όταν υποστηρίζεται η **διοχέτευση (pipeline)**
 - Σε κάθε βήμα να εκτελείται διαφορετική εντολή

Επιβάρυνση δημιουργίας ακολουθίας = καθυστέρηση διάδοσης από το CLK έως την έξοδο Q του καταχωρητή + χρόνος σταθεροποίησης καταχωρητή.

Τελικά, η επιβάρυνση δημιουργίας ακολουθίας είναι ένας πολύ κρίσιμος παράγοντας γιατί είναι μία σημαντική καθυστέρηση διάδοσης ανάλογη ενός γρήγορου αθροιστή των 32-bit

Επεξεργαστές πολλών κύκλων είχαν τα υπολογιστικά συστήματα IBM 360, DEC VAX και οι μικρο-επεξεργαστές 6502 και 8088. Υπάρχουν ακόμα ως φθινοί μικροελεγκτές για οικιακές συσκευές και παιχνίδια. Αντικαθίστανται από επεξεργαστές με διοχέτευση. Η Intel εισήγαγε την διοχέτευση από τον 80486 το 1989. Όλοι οι RISC υποστηρίζουν σήμερα διοχέτευση.

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

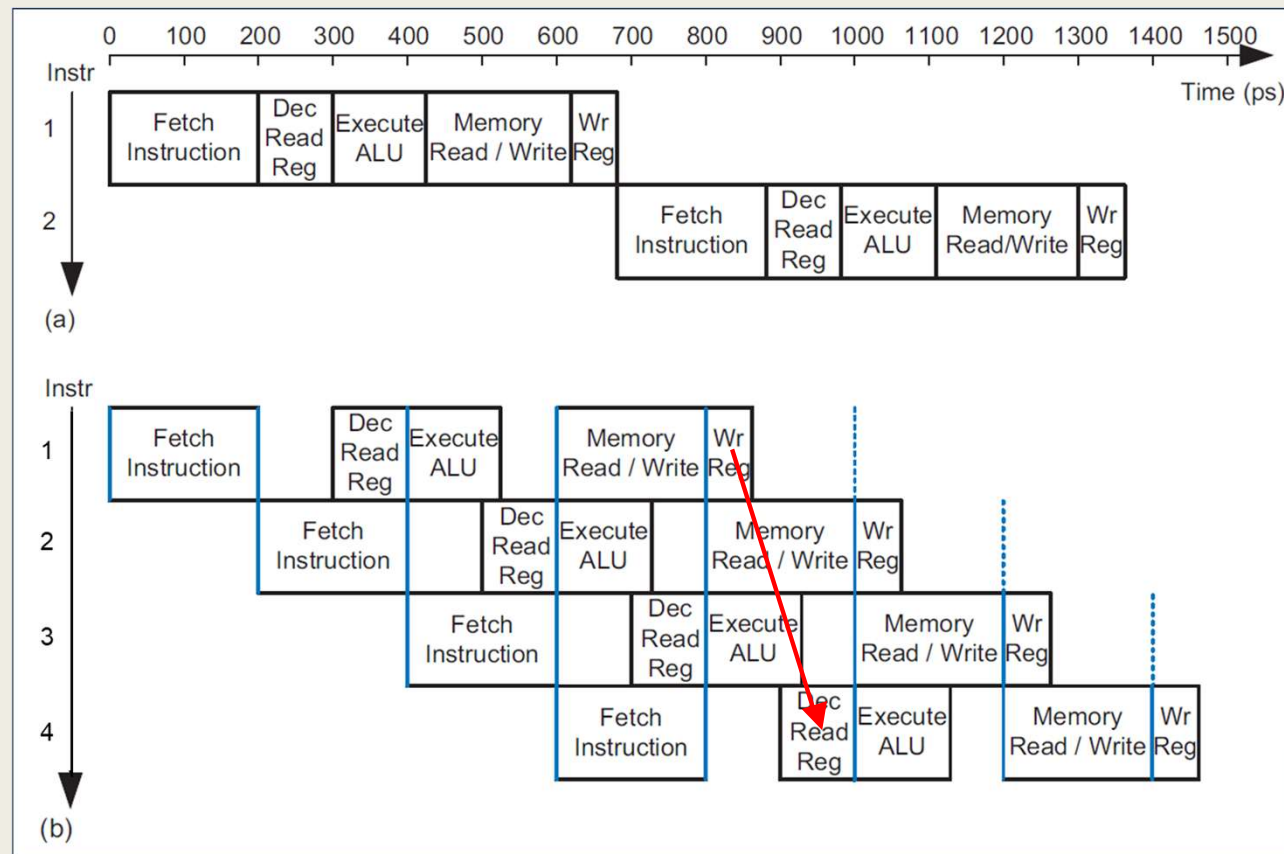
- Θα σχεδιάσουμε έναν επεξεργαστή πολλών κύκλων με διοχέτευση χωρίζοντας τον επεξεργαστή ενός κύκλου σε **πέντε στάδια διοχέτευσης**:
 1. **FETCH**: Προσκόμιση εντολής και υπολογισμός διεύθυνσης και πρόβλεψη της επόμενης εντολής (PC+4)
 2. **DECODE**: Αποκωδικοποίηση εντολής, ανάγνωση αρχείου καταχωρητών και υπολογισμός άμεσου τελεστέου των 32 bit
 3. **EXECUTE**: Εκτέλεση πράξεων στη μονάδα ALU και στον αθροιστή PCtarget, και επιλογή της διεύθυνσης της επόμενης εντολής
 4. **MEMORY**: Ανάγνωση ή εγγραφή στη μνήμη δεδομένων
 5. **WRITEBACK**: Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών
- Συνεκτιμώντας ότι η υλοποίηση θα γίνει σε τεχνολογία FPGA προβαίνουμε στις ακόλουθες αρχιτεκτονικές επιλογές για την υλοποίηση της διοχέτευσης:
 - η **μνήμη εντολών** παραμένει **χωριστά** από τη **μνήμη δεδομένων**
 - διατηρούνται οι **δύο αθροιστές PCplus4** και **Pctarget**
 - τοποθετούνται 4 **καταχωρητές διοχέτευσης (Pipeline Registers- PL)** ως **μη αρχιτεκτονικοί καταχωρητές** στην διαδρομή δεδομένων και στην μονάδα ελέγχου
 - για τον διαχωρισμό των πέντε σταδίων διοχέτευσης, αξιοποιώντας διαθέσιμα D-Flip-flops
 - Ο επεξεργαστής λειτουργεί με **πρόβλεψη εκτέλεσης των 2 επόμενων κατά σειρά εντολών** (και απόρριψη αυτών, όπου απαιτείται)
 - Όλες οι εντολές εκτελούνται σε **5 κύκλους**, εκτός να υπάρχει απόρριψη

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

- Ο επεξεργαστής πολλών κύκλων με διοχέτευση των πέντε σταδίων δύναται να εκτελέσει πέντε εντολές ταυτόχρονα, αφού μία διαφορετική εντολή εκτελείται σε κάθε στάδιο στο ίδιο κύκλο ρολογιού
 - Όλες οι εντολές εκτελούνται σε 5 κύκλους ρολογιού ταυτόχρονα διερχόμενες και από τα πέντε στάδια η μία μετά την άλλη στην σειρά
 - Ιδανικά, ο λανθάνων χρόνος κάθε εντολής θα είχε μία μικρή επιβάρυνση (4 φορές την επιβάρυνση δημιουργίας ακολουθίας), αλλά η διεκπεραιωτική ικανότητα θα γινόταν σχεδόν πενταπλάσια.
 - Απαιτείται διαχείριση κινδύνων λόγω εξαρτήσεων δεδομένων και ελέγχου από την μία εντολή στην άλλη με την σχεδίαση της μονάδας διαχείρισης κινδύνων (**Hazard Unit**)
 - Η αναγκαιότητα για εκτέλεση μίας διαφορετικής εντολής σε κάθε στάδιο απαιτεί:
 - όλα τα δεδομένα και τα σήματα ελέγχου να αποθηκεύονται σε 4 ενδιάμεσους καταχωρητές διοχέτευσης (όταν χρησιμοποιούνται σε δύο διαδοχικά στάδια, η αποθήκευση γίνεται στον ενδιάμεσο καταχωρητή διοχέτευσης)
 - στο 1^ο στάδιο προεπιλέγεται ως επόμενη τιμή του PC, το PC+4
 - Το αρχείο καταχωρητών εγγράφει δεδομένα κατά την κατερχόμενη ακμή του CLK
 - ώστε να μπορεί να εγγράψει ένα αποτέλεσμα στο πρώτο μισό ενός κύκλου και να δύναται να διαβάσει το ίδιο αποτέλεσμα στο δεύτερο μισό του ιδίου κύκλου στην 3^η κατά σειρά επόμενη εντολή.

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

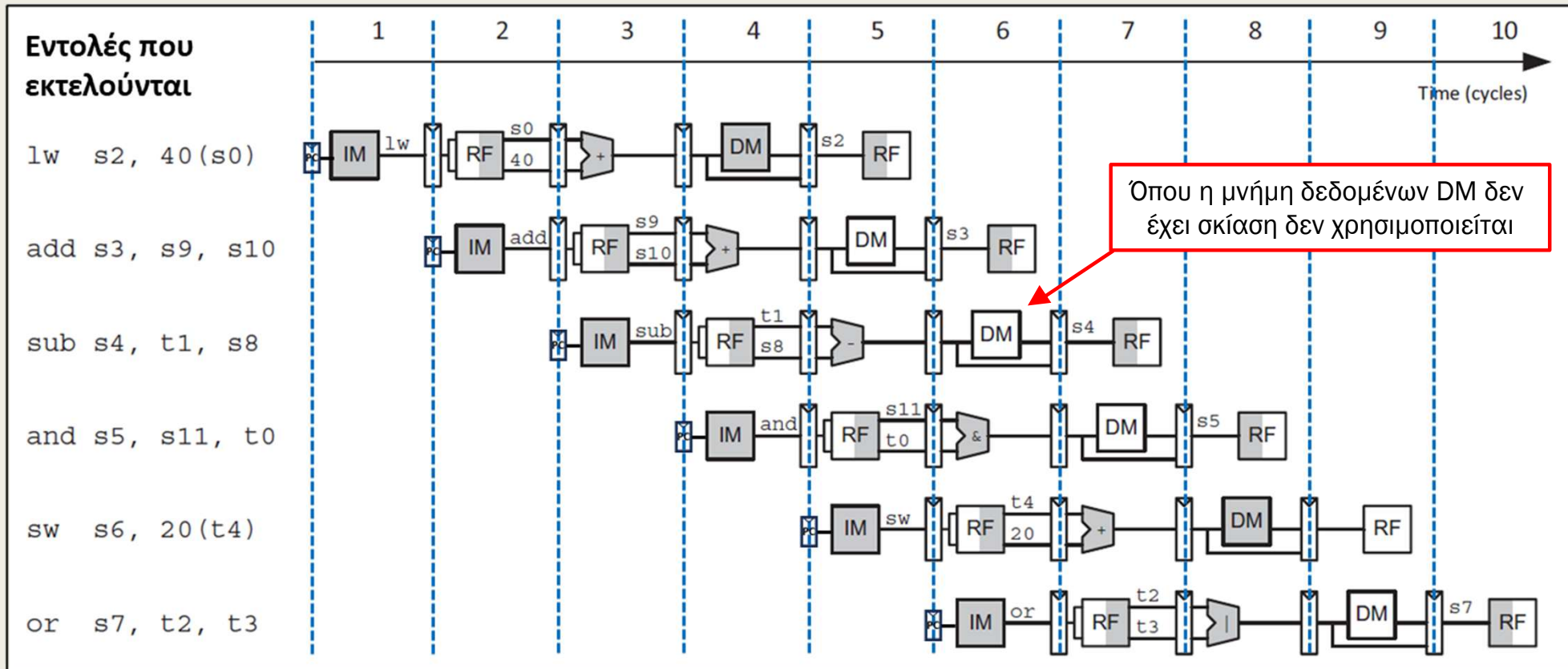
- Παράδειγμα διαγράμματος χρονισμού στο οποίο γίνεται σύγκριση (α) του επεξεργαστή ενός κύκλου και (β) του επεξεργαστή με διοχέτευση
 - (α) επεξεργαστής ενός κύκλου: λανθάνων χρόνος = 680 ps, διεκπεραιωτική ικανότητα = $1/680 \text{ ps} = 1,47 \text{ Ginst/s}$
 - (β) επεξεργαστής με διοχέτευση: λανθάνων χρόνος = $200 \times 5 = 1.000 \text{ ps}$, διεκπεραιωτική ικανότητα = $1/200 \text{ ps} = 5 \text{ Ginst/s}$ (x3,4)



Κάθε στάδιο διοχέτευσης υποδεικνύεται με συνεχή ή διακεκομμένη κατακόρυφη μπλε γραμμή

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

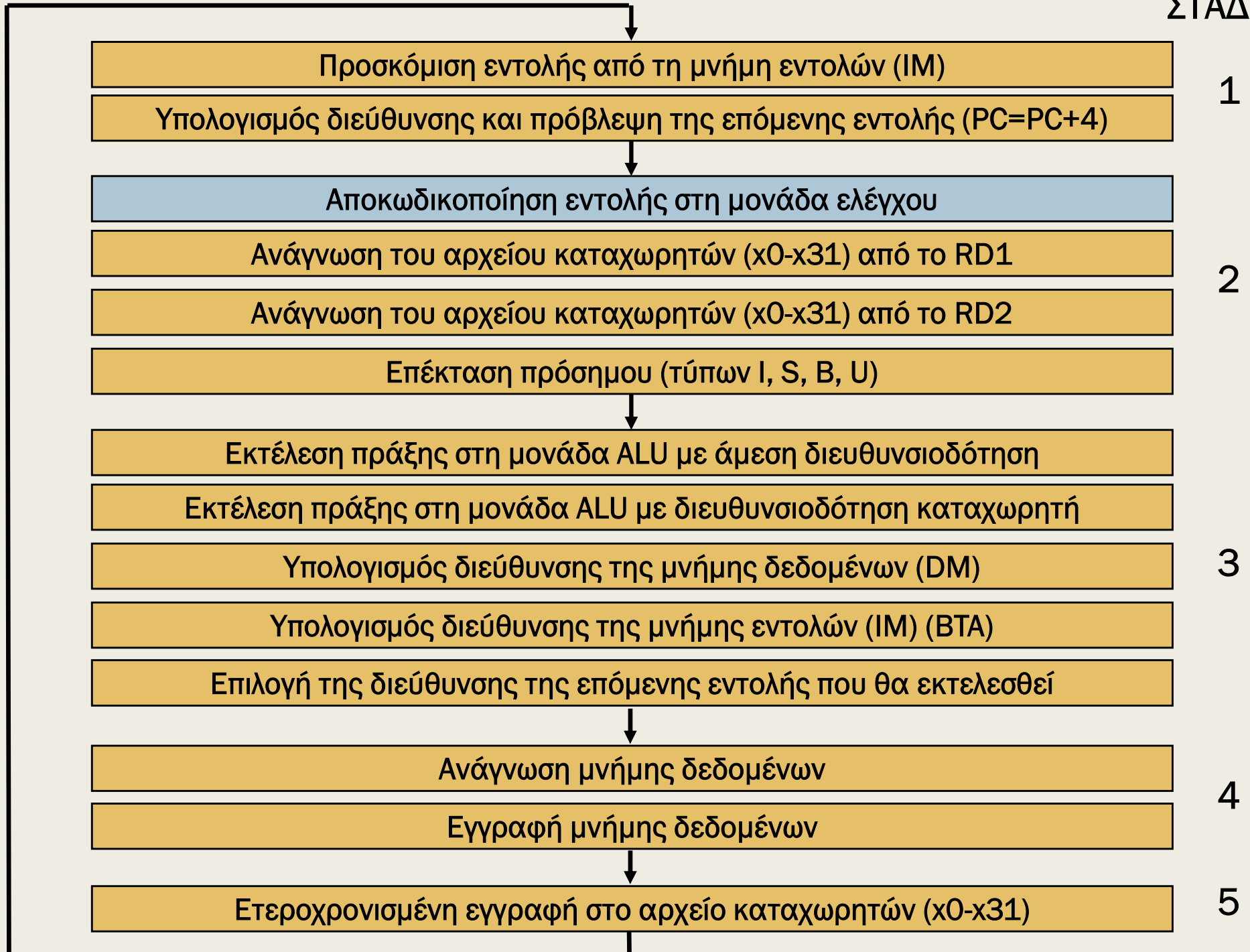
- Αφηρημένη θεώρηση της διοχέτευσης σε λειτουργία, όπου κάθε στάδιο αναπαρίσταται σχηματικά



Κάθε στάδιο της διοχέτευσης αναπαρίσταται με το κύριο επιμέρους στοιχείο του –μνήμη εντολών (instruction memory, IM), ανάγνωση από αρχείο καταχωρητών (register file, RF) στο δεύτερο μισό του κύκλου, εκτέλεση πράξης στη μονάδα ALU (που υποδεικνύεται στο διάγραμμα με το σύμβολο της αντίστοιχης πράξης), μνήμη δεδομένων (data memory, DM), και ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών (register file, RF) στο πρώτο μισό του κύκλου– ώστε να φαίνεται η ροή των δεδομένων μέσω της διοχέτευσης. Τα στάδια που χρησιμοποιούνται υποδεικνύονται με σκίαση.

Λειτουργίες των βασικών εντολών (14)

ΣΤΑΔΙΑ

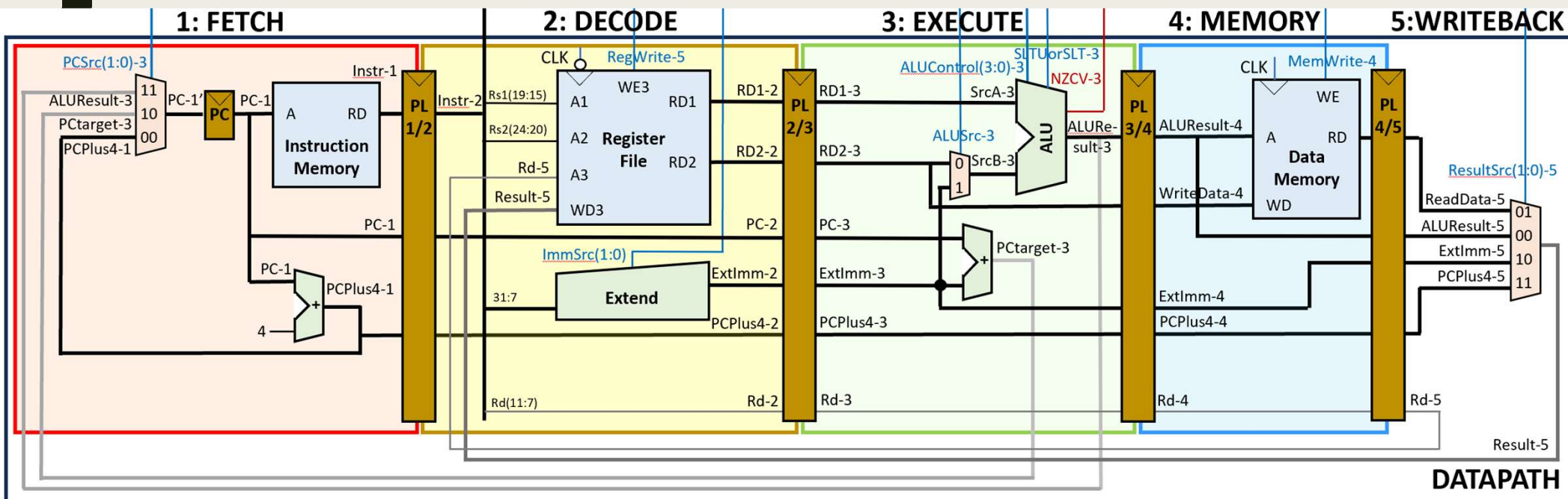


Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

- Κατά τη σχεδίαση προβαίνουμε σε:

- μελέτη διαδρομής δεδομένων ανά στάδιο εκτέλεσης της εντολής
- μελέτη μονάδας ελέγχου
- ανάλυση επιδόσεων

- Μετά την μελέτη της διαδρομής δεδομένων και την τοποθέτηση των απαιτούμενων 4 καταχωρητών διοχέτευσης (PL) η διαδρομή δεδομένων του επεξεργαστή πολλών κυκλών με υποστήριξη διοχέτευσης 5 σταδίων είναι η ακόλουθη:



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

- Τα δεδομένα αποθηκεύονται σε 4 ενδιάμεσους **καταχωρητές διοχέτευσης** για την εκτέλεση όλων των βασικών εντολών
 - όταν χρησιμοποιούνται σε δύο διαδοχικά στάδια, η αποθήκευση γίνεται στον ενδιάμεσο καταχωρητή διοχέτευσης μεταξύ αυτών των δύο σταδίων

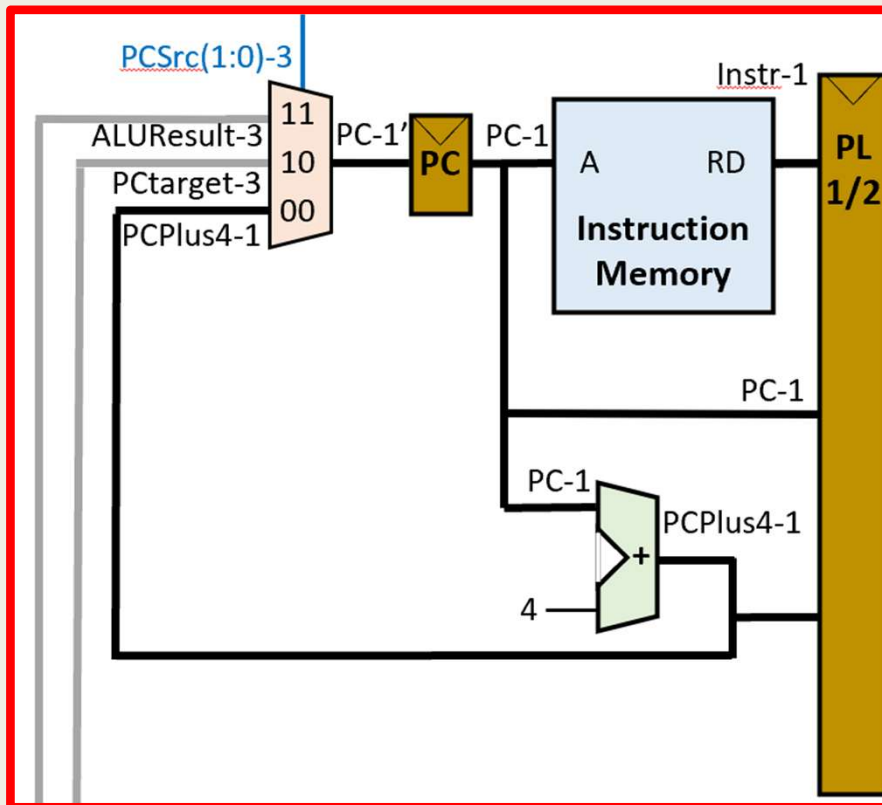
Δεδομένα	1.FETCH	2.DECODE	3.EXECUTE	4.MEMORY	5.WRITEBACK
PC	PC-1	PC-2	PC-3		
PCplus4 (RF)	PCplus4-1	PCplus4-2	PCplus4-3	PCplus4-4	PCplus4-5
PCplus4 (PC)	PCplus4-1				
Instr (31:0)	Instr-1	Instr-2			
RD1		RD1-2	RD1-3		
RD2		RD2-2	RD2-3	WriteData-4	
Extimm (RF)		Extimm-2	Extimm-3	Extimm-4	Extimm-5
Rd(11:7)		Rd-2	Rd-3	Rd-4	Rd-5
ALUResult (RF)			ALUResult-3	ALUResult-4	ALUResult-5
ALUResult (PC)			ALUResult-3		
PCtarget (PC)			PCtarget-3		
NZCV (to Control)			NZCV-3		
ReadData (RF)				ReadData-4	ReadData-5

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

- Η διάδοση των δεδομένων μέσα στην διαδρομή δεδομένων παρουσιάζει τις εξής ιδιαιτερότητες:
 - Το αρχείο καταχωρητών είναι ιδιόμορφο επειδή διαβάζεται στο στάδιο 2.DECODE και εγγράφεται στο στάδιο 5.WRITEBACK
 - Η καταχωρητής προορισμού Rd (instr(11:7)) που προσδιορίζεται στο στάδιο 2.DECODE, καθώς και τα δεδομένα εγγραφής (PCplus4, Extimm, ALUResult ή ReadData) πρέπει να διαδοθούν κατάλληλα μέσω των καταχωρητών διοχέτευσης, ώστε να είναι διαθέσιμα στο στάδιο 5.WRITEBACK ως Rd-5, PCplus4-5, Extimm-5, ALUResult-5 και ReadData-5
 - Ο επεξεργαστής λειτουργεί με **πρόβλεψη εκτέλεσης των 2 επόμενων κατά σειρά εντολών**
 - Η επόμενη τιμή του PC, $PC-1' = PCplus4-1 = PC+4$ παράγεται στο στάδιο 1.FETCH για να επιτευχθεί η πρόβλεψη
 - Η πρόβλεψη ισχύει για όλες τις εντολές, εκτός από τις εντολές Branch (στην περίπτωση που ικανοποιείται η συνθήκη) και την εντολή JALR/JR, όπου απαιτείται **απόρριψη**
 - Ο πολυπλέκτης επιλογής της επόμενης τιμής του PC ($PC-1'$) χρησιμοποιείται στο στάδιο 1.FETCH για την επιλογή της αμέσως επόμενης εντολής στην διεύθυνση $PC+4$, ως πρόβλεψη, αλλά επιλέγει την ορθή επόμενη τιμή του PC στο σταδίο 3.EXECUTE μεταξύ των διαθέσιμων διευθύνσεων επόμενων εντολών:
 - $PCplus4-1 = PC+4$ (που έχει ήδη προεπιλεγεί στο στάδιο 1.FETCH – καμία ενέργεια)
 - $ALUResult-3 = RF(Rs1) + Sign_Ext(instr(31:20))$
(αποκλειστικά για τις εντολές JALR, που παράγεται στο στάδιο 3.EXECUTE – απαιτείται απόρριψη)
 - $PCtarget-3 = BTA$
(αποκλειστικά για τις εντολές Branch (όταν ικανοποιείται η συνθήκη, που παράγεται στο στάδιο 3.EXECUTE – απαιτείται απόρριψη)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

1. **FETCH**: Προσκόμιση εντολής και υπολογισμός διεύθυνσης και πρόβλεψη της επόμενης εντολής (PC+4)



Λειτουργίες

$\text{Instr-1} = \text{IM}(\text{PC-1})$

$\text{PCPlus4-1} = \text{PC-1} + 4$

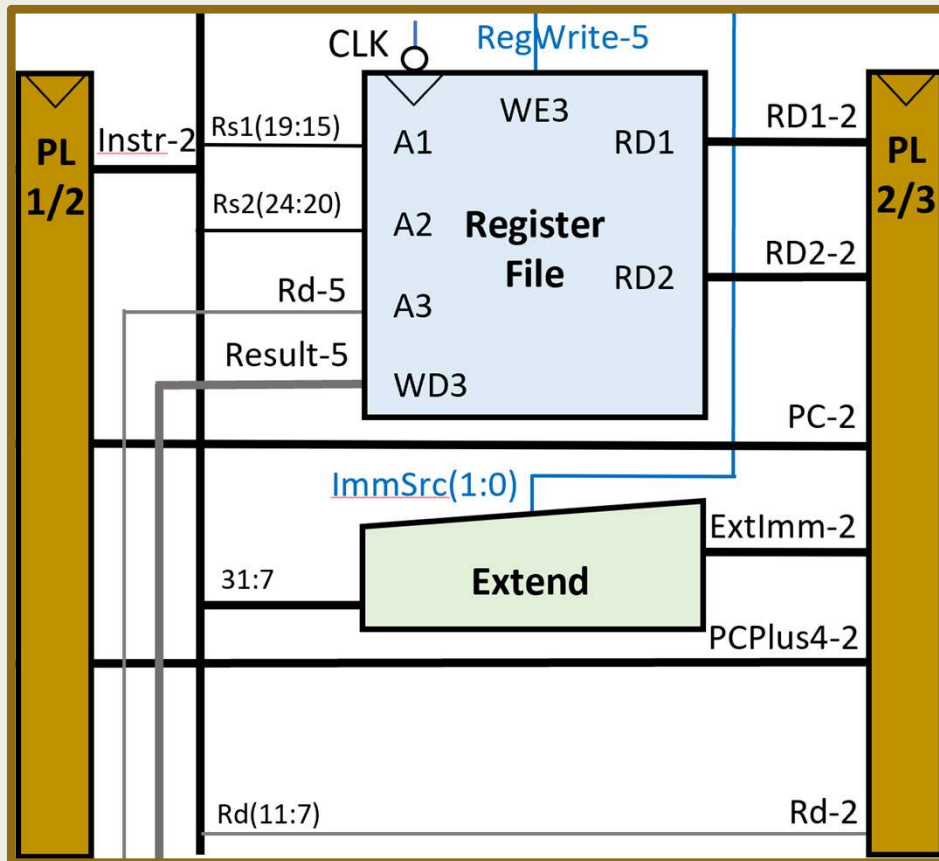
$\text{PC-1}' = \text{PCPlus4-1}$ (πρόβλεψη)

Ο συγκεκριμένος επεξεργαστής λειτουργεί με πρόβλεψη εκτέλεσης των δύο επόμενων κατά σειρά εντολών στις διευθύνσεις PC+4 και PC+8, αντίστοιχα.

Το **PCSrc(1:0)-3 = 00** μέχρι να εκτελεσθεί στο 3^ο στάδιο μία εντολή Branch, που ικανοποιεί την συνθήκη, ή μία εντολή JALR, οπότε γίνεται απόρριψη αυτών των δύο εντολών (**PCSrc(1:0)-3 ≠ 00**)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

2. **DECODE**: Αποκωδικοποίηση εντολής, ανάγνωση αρχείου καταχωρητών και υπολογισμός άμεσου τελεστέου των 32 bit



Λειτουργίες

$RD1-2 = RF(instr-2(Rs1))$ – Ασύγχρονα

$RD2-2 = RF(instr-2(Rs2))$ – Ασύγχρονα

$ExtImm-2 = Sign_Ext(instr-2(31:20))$

– **ImmScr(1:0) = 00 (I)**

$ExtImm-2 = Sign_Ext(instr-2(31:25) \& instr-2(11:7))$

– **ImmScr(1:0) = 01 (S)**

$ExtImm-2 = Sign_Ext(instr-2(31) \& instr-2(7) \& instr-2(30:25) \& instr-2(11:8) \& '0')$

– **ImmScr(1:0) = 10 (B)**

$ExtImm-2 = instr-2(31:12) \& 0x000$

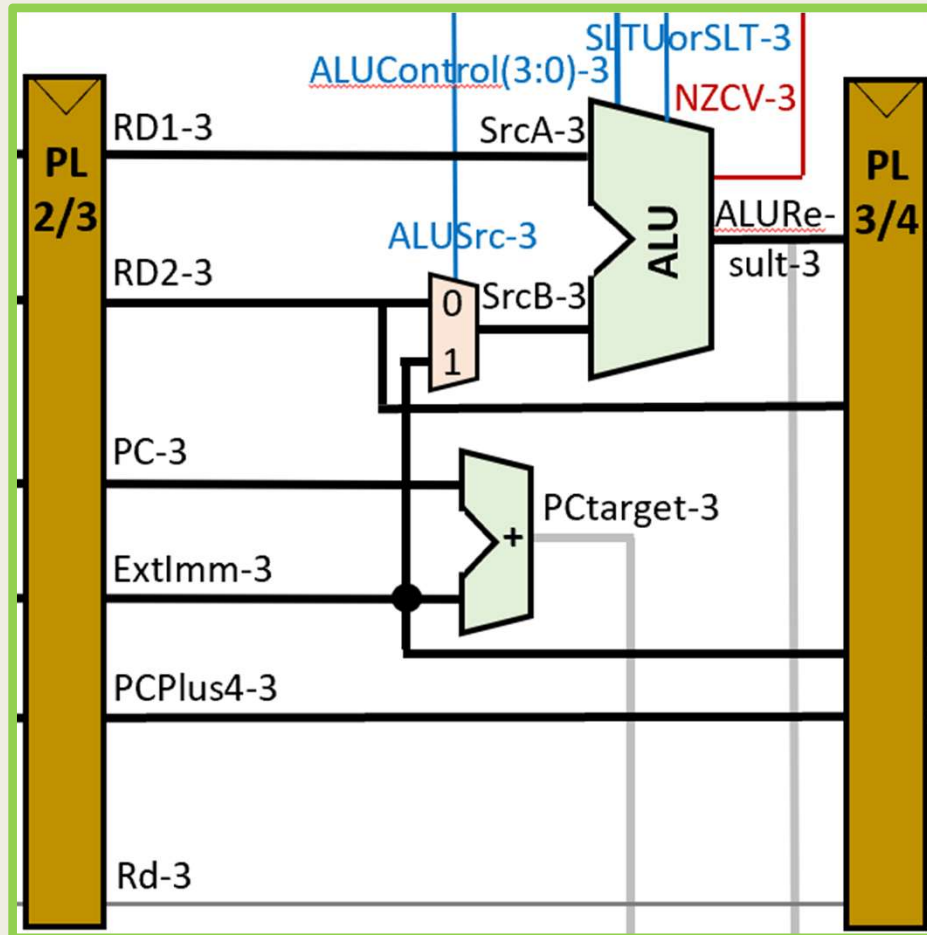
– **ImmScr(1:0) = 11 (U)**

11	10	9	8	7	6	5	4	3	2	1	0	rs1	funct3	rd	I									
11	10	9	8	7	6	5	rs2					rs1	funct3	4	3	2	1	0	S					
12	10	9	8	7	6	5	rs2					rs1	funct3	4	3	2	1	11	B					
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	rd	U			
20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	rd	J		
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7

Γίνεται η ανάγνωση από το RF
στο **δεύτερο μισό του κύκλου**

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

3. **EXECUTE**: Εκτέλεση πράξεων στη μονάδα ALU και στον αθροιστή PCtarget, καθώς και επιλογή της διεύθυνσης της επόμενης εντολής



ALUControl(3:0)	Πράξη
0000	Add
0001	Subtract
0011 (SLTUorSLT=0)	SLT
0011 (SLTUorSLT=1)	SLTU

0100	AND
0101	OR
011X	XOR
1X0X	SLL
1X10	SRL
1X11	SRA

Λειτουργίες

ALUResult-3 = RD1-3 +/- and/or/xor RD2-3
(ADD, SUB, AND, OR, XOR) - **ALUSrc-3 = 0**

ALUResult-3 = 1, if RD1-3 < RD2-3, else 0 (SLT/SLTU)
- **ALUSrc-3 = 0, SLTUorSLT-3 = 0 (S) / 1 (U)**

ALUResult-3 = RD1-3 <</>>/ >>> RD2-3(4:0)
(SLL, SRL, SRA) - **ALUSrc-3 = 0**

Z = 1, if RD1-3 = RD2-3, else 0 (BEQ/BNE)
- **ALUSrc-3 = 0**

ALUResult-3 = RD1-3 +/-and/or/xor Extimm-3
(LW, ADDI, ANDI, ORI, XORI)
- **ALUSrc-3 = 1, ImmScr(1:0) = 00 (I)**

ALUResult-3 = RD1-3 + Extimm-3 (JALR/JR)
- **ALUSrc-3 = 1, ImmScr(1:0) = 00 (I)**

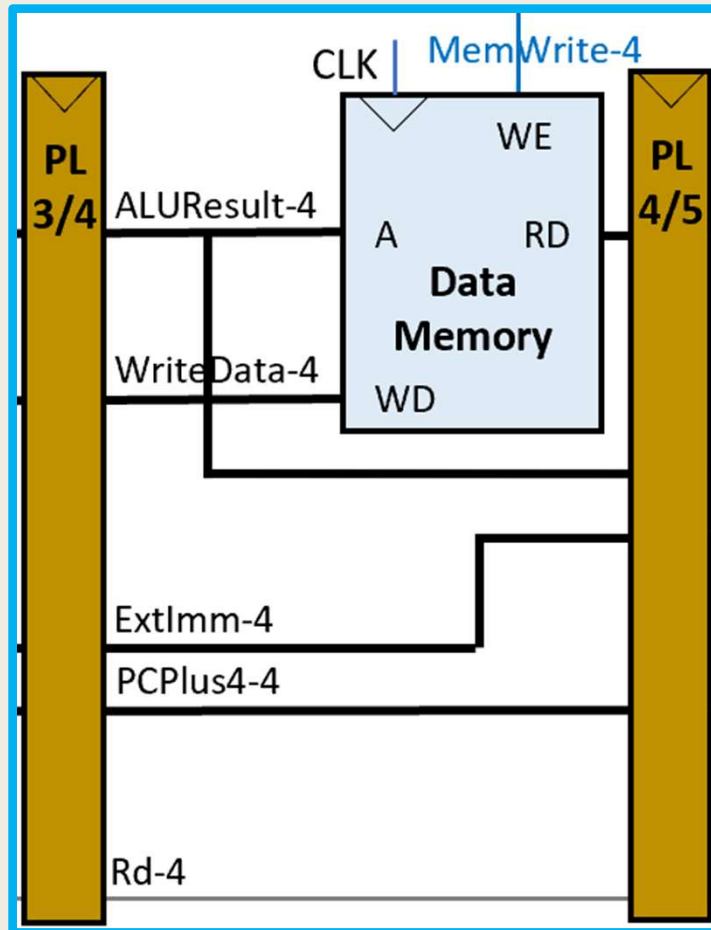
ALUResult-3 = RD1-3 + Extimm-3 (SW)
- **ALUSrc-3 = 1, ImmScr(1:0) = 01 (S)**

PCtarget-3 = PC-3 + Extimm-3 (BTA, BEQ/BNE)
- **ImmScr(1:0) = 10 (B)**

- **PCSrc(1:0)-3 = 00** (Όλες οι εντολές, εκτός Branch/JALR)
- **PCSrc(1:0)-3 = 00** (BEQ & Z=0, BNE & Z=1)
- **PCSrc(1:0)-3 = 10** (BEQ & Z=1, BNE & Z=0)
- **PCSrc(1:0)-3 = 11** (JALR/JR)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

4. **MEMORY**: Ανάγνωση ή εγγραφή στη μνήμη δεδομένων



Λειτουργίες

ReadData-4 = DM(ALUResult-4) (LW)

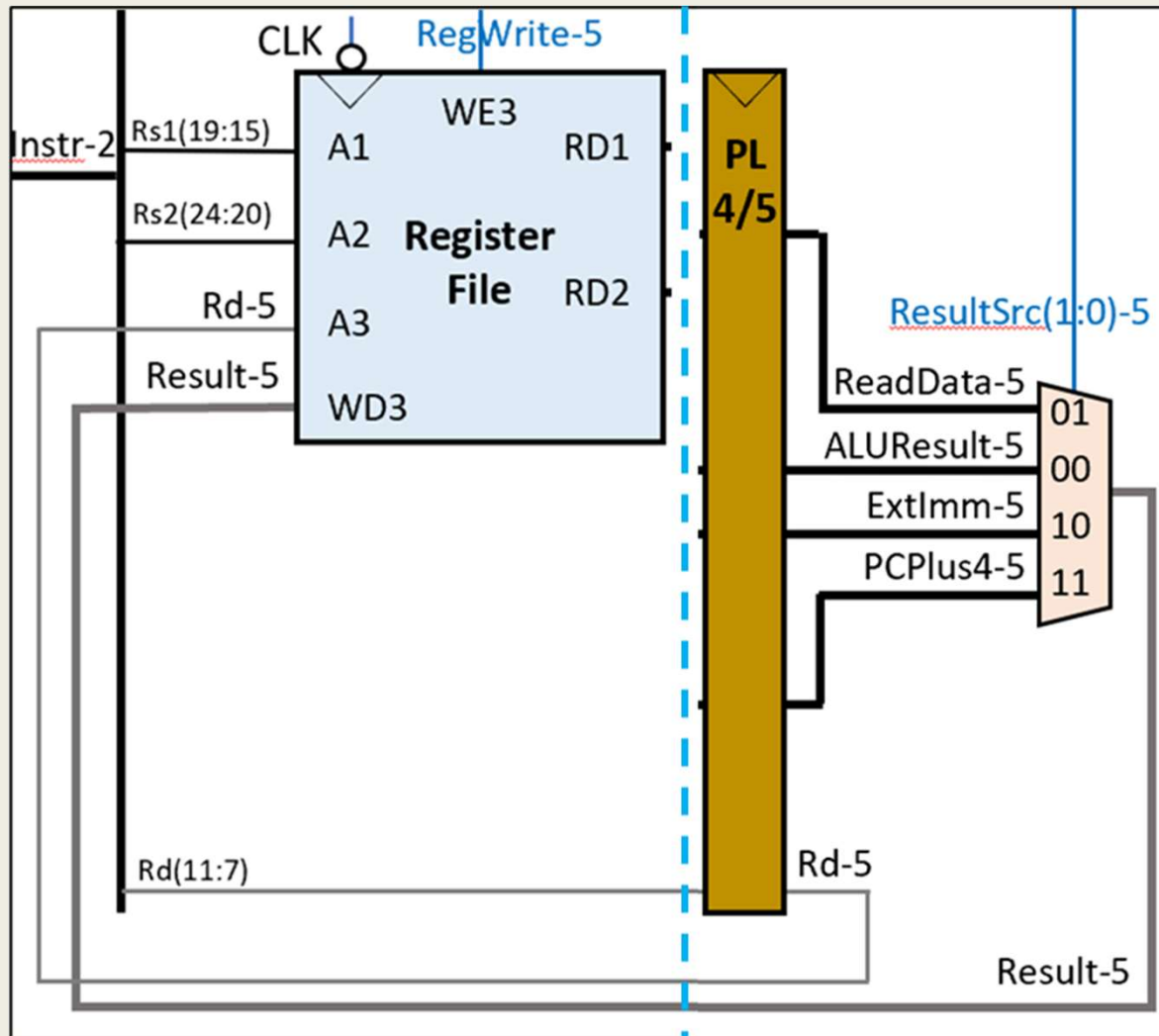
- **MemWrite = 0**

DM(ALUResult-4) = WriteData-4 (SW)

- **MemWrite = 1**

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): διαδρομή δεδομένων

5. **WRITEBACK**: Ετεροχρονισμένη εγγραφή στο αρχείο καταχωρητών



Λειτουργίες

RF(Rd-5) = ReadData-5 (LW)

- RegWrite = 1

$$\text{RF(Rd-5)} = \text{ALUResult-5 (ALU/ALUI)}$$

- RegWrite = 1

RF(Rd-5) = Extimm-5 (LUI)

- RegWrite = 1, ImmScr(1:0) = 11 (U)

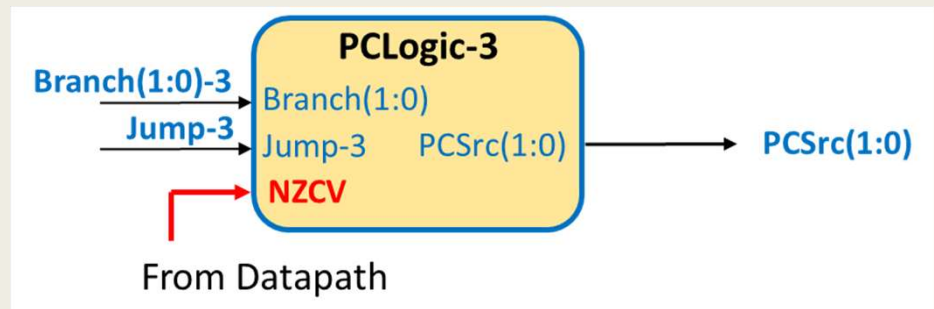
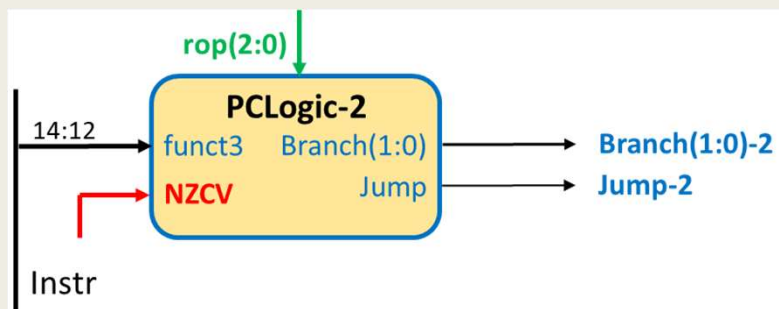
RF(Rd-5) = PCPlus4-5 (JAL)

- RegWrite = 1

Γίνεται η εγγραφή στο RF
στο πρώτο μισό του κύκλου

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

- Στη συνέχεια της σχεδίασης προβαίνουμε σε μελέτη της **μονάδας ελέγχου**
- Ο επεξεργαστής με διοχέτευση χρησιμοποιεί τα ίδια σήματα ελέγχου, όπως ο επεξεργαστής ενός κύκλου και, άρα, την ίδια μονάδα ελέγχου.
 - Αυτά τα σήματα ελέγχου πρέπει να διοχετεύονται μαζί με τα δεδομένα, ώστε να παραμένουν συγχρονισμένα σε όλα τα στάδια εκτέλεσης της εντολής.
- Η μόνη εξαίρεση είναι η υλοποίηση της υπομονάδας επιλογής διεύθυνσης επόμενης εντολής PCLogic(1:0), η οποία υλοποιείται σε δύο διακριτές υπομονάδες PCLogic-2 και PCLogic-3 που επικοινωνούν μέσω του καταχωρητή διοχέτευσης PL2/3:
 - στο στάδιο 2.DECODE η υπομονάδα PCLogic-2 λαμβάνει το εσωτερικό σήμα **rop(2:0)** και το πεδίο **funct3** (Instr(14:12) και παράγει τα εσωτερικά σήματα **Branch-2** και **Jump-2**
 - στο στάδιο 3.EXECUTE η υπομονάδα PCLogic-3 λαμβάνει τα εσωτερικά σήματα **Branch-3** και **Jump-3** και παράγει το ορθό σήμα **PCSrc(1:0)** σύμφωνα με τα ALU flags **NZCV-3** που παράγονται στην μονάδα ALU στο ίδιο στάδιο



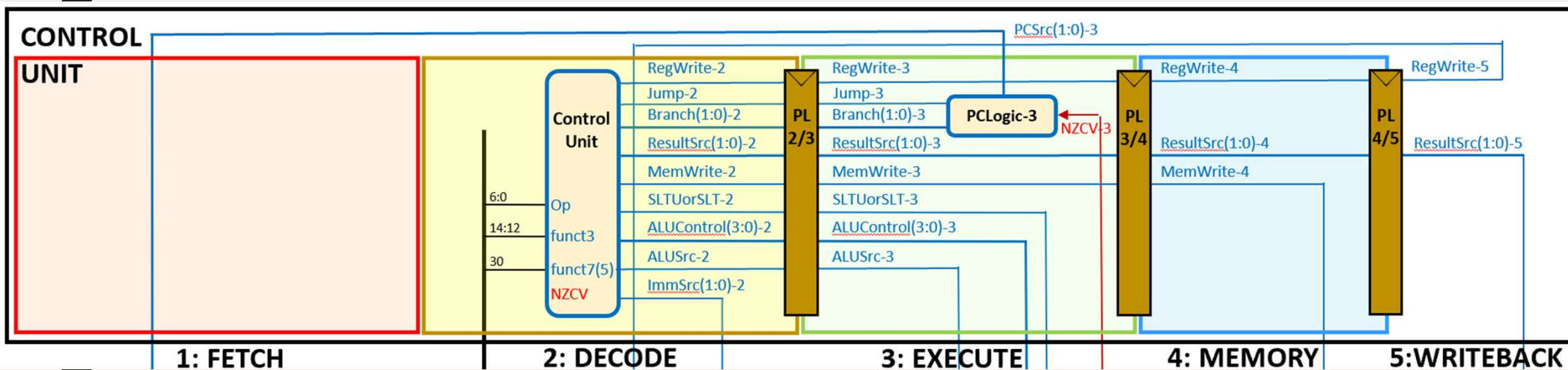
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

3. Κατανεμημένη λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic-2 και PCLogic-3)

Εντολή	op(6:0)	rop(2:0)	funct3	Branch(1:0)-2/3	Jump-2/3	NZCV-3	PCSrc(1:0)-3
LW	00000	000	XXX	00	0	XXXX	00
ALUI	00100	001	XXX	00	0	XXXX	00
SW	01000	010	XXX	00	0	XXXX	00
ALU	01100	011	XXX	00	0	XXXX	00
BEQ	11000	100	000	10	0	X0XX	00
BEQ	11000	100	000	10	0	X1XX	10
BNE	11000	100	001	11	0	X0XX	10
BNE	11000	100	001	11	0	X1XX	00
JALR	11001	101	XXX	00	1	XXXX	11
LUI	01101	11X	XXX	00	0	XXXX	00

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

- Μετά την μελέτη της μονάδας ελέγχου και την τοποθέτηση των απαιτούμενων 3 καταχωρητών διοχέτευσης (PL) η μονάδα ελέγχου του επεξεργαστή πολλών κυκλών με υποστήριξη διοχέτευσης 5 σταδίων είναι η ακόλουθη:



Η υπομονάδα PCLogic-2 είναι ενσωματωμένη στην Control Unit

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

- Τα σήματα ελέγχου αποθηκεύονται σε 3 ενδιάμεσους **καταχωρητές διοχέτευσης** για την εκτέλεση όλων των βασικών εντολών
 - όταν χρησιμοποιούνται σε δύο διαδοχικά στάδια, η αποθήκευση γίνεται στον ενδιάμεσο καταχωρητή διοχέτευσης μεταξύ αυτών των δύο σταδίων

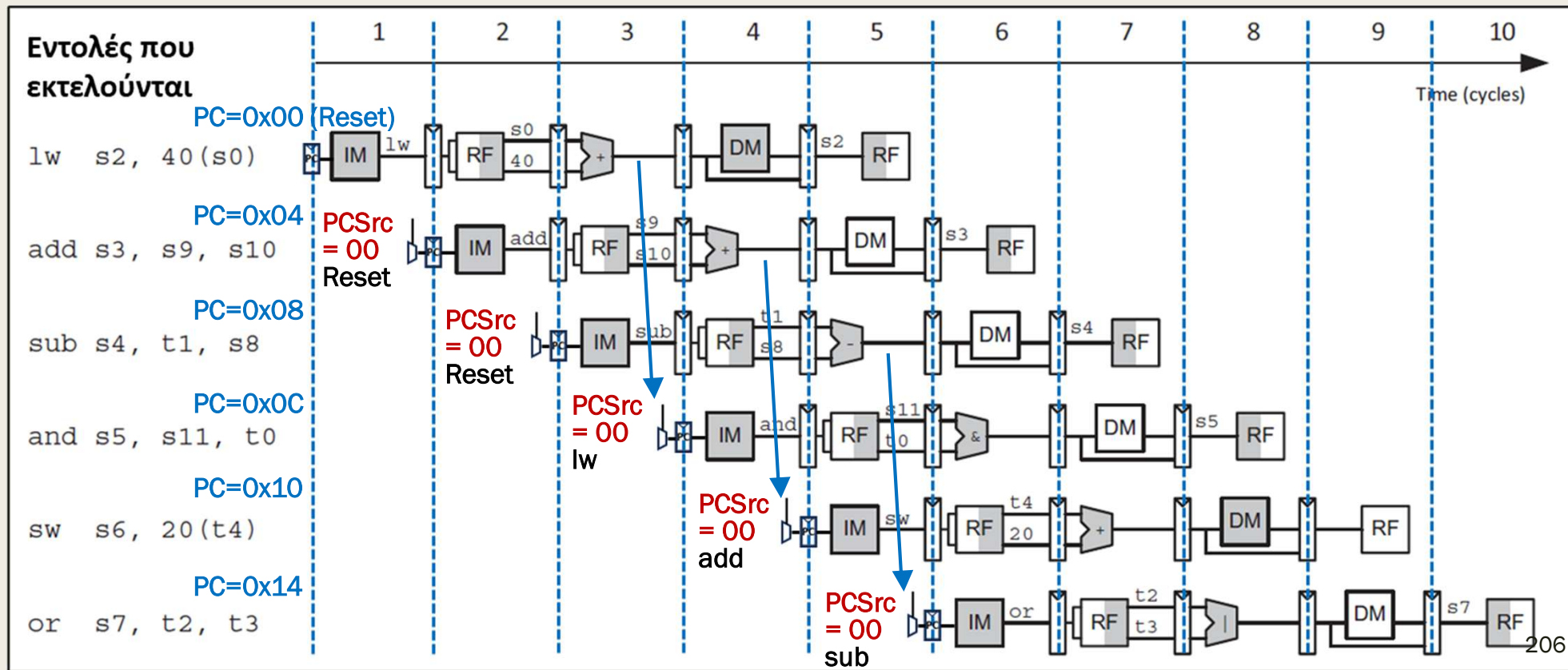
Σήματα ελέγχου (CU)	2.DECODE	3.EXECUTE	4.MEMORY	5.WRITEBACK
NZCV (from datapath)		NZCV-3		
ImmSrc(1:0)	ImmSrc(1:0)-2			
ALUSrc	ALUSrc-2	ALUSrc-3		
ALUControl(3:0)	ALUControl(3:0)-2	ALUControl(3:0)-3		
SLTUorSLT	SLTUorSLT-2	SLTUorSLT-3		
MemWrite	MemWrite-2	MemWrite-3	MemWrite-4	
ResultSrc(1:0)	ResultSrc(1:0)-2	ResultSrc(1:0)-3	ResultSrc(1:0)-4	ResultSrc(1:0)-5
RegWrite	RegWrite-2	RegWrite-3	RegWrite-4	RegWrite-5
PCSrc(1:0)	Branch(1:0)-2	Branch(1:0)-3		
	Jump-2	Jump-3		
		PCSrc(1:0)-3		

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

- Η διάδοση των σημάτων ελέγχου μέσα στην διαδρομή δεδομένων παρουσιάζει τις εξής ιδιαιτερότητες:
 - *Πρόβλεψη εκτέλεσης των 2 επόμενων κατά σειρά εντολών: Ανεξάρτητα εάν πρόκειται να εκτελεσθεί μία εντολή Branch ή JALR/JR στην διεύθυνση PC, για κάθε εντολή στη διεύθυνση PC προβλέπεται στους επόμενους δύο κύκλους να ξεκινήσει η εκτέλεση των δύο επόμενων κατά σειρά εντολών στις διευθύνσεις PC+4 και PC+8, αντίστοιχα*
 - *Όταν η εντολή στην διεύθυνση PC φθάσει στο στάδιο 3.Execute (3^{ος} κύκλος) παράγεται η πραγματική τιμή του σήματος **PCSrc(1:0)-3**
 - **PCSrc(1:0)-3 = “00”** (για όλες τις εντολές, εκτός Branch και JALR)
 - **PCSrc(1:0)-3 = “00”** (για τις εντολές Branch όταν δεν ικανοποιείται η συνθήκη)
 - **PCSrc(1:0)-3 = “10”** (για τις εντολές Branch όταν ικανοποιείται η συνθήκη)
 - **PCSrc(1:0)-3 = “11”** (για τις εντολές JALR/JR)*
 - *Εάν **PCSrc(1:0)-3 = ‘1’** απαιτείται **απόρριψη αυτών των 2 εντολών**, που άρχισαν να εκτελούνται στις διευθύνσεις PC+4 και PC+8, αντίστοιχα, λόγω εσφαλμένης προβλέψεως και παρουσίας **κινδύνου ελέγχου (control hazard)***
 - Η διαδικασία της απόρριψης θα μελετηθεί στη συνέχεια

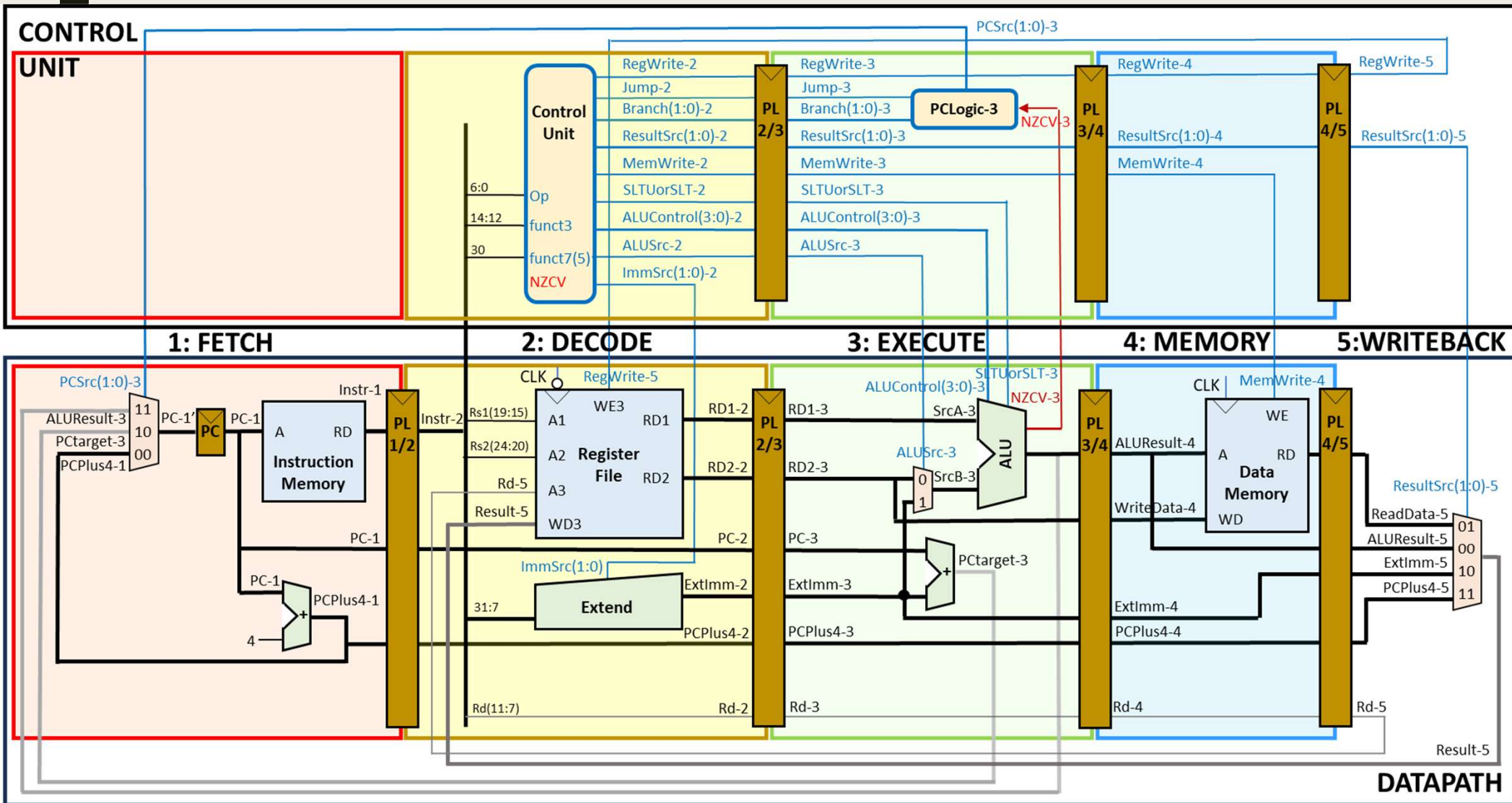
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα ελέγχου

- Πρόβλεψη εκτέλεσης των 2 επόμενων κατά σειρά εντολών: Για κάθε εντολή στη διεύθυνση PC προβλέπεται στους επόμενους δύο κύκλους να ξεκινήσει η εκτέλεση των δύο επόμενων κατά σειρά εντολών στις διευθύνσεις PC+4 και PC+8, αντίστοιχα
 - Στην αρχή της λειτουργίας του επεξεργαστή, το Global Reset σε όλους τους καταχωρητές καθορίζει $PC=0$, καθώς και $PCSrc(1:0)-2 = "00"$ και $PCSrc(1:0)-3 = 00"$
 - Στους κύκλους 1 και 2 το $PCSrc(1:0)-3 = "00"$ λόγω Global Reset
 - Στους επόμενους κύκλους η τιμή του $PCSrc(1:0)-3$ εξαρτάται από την εντολή που βρίσκεται στο στάδιο 3.EXECUTE (3 εντολές πίσω)



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining)

- Ολοκληρωμένος επεξεργαστής πολλών κύκλων με απλή διοχέτευση



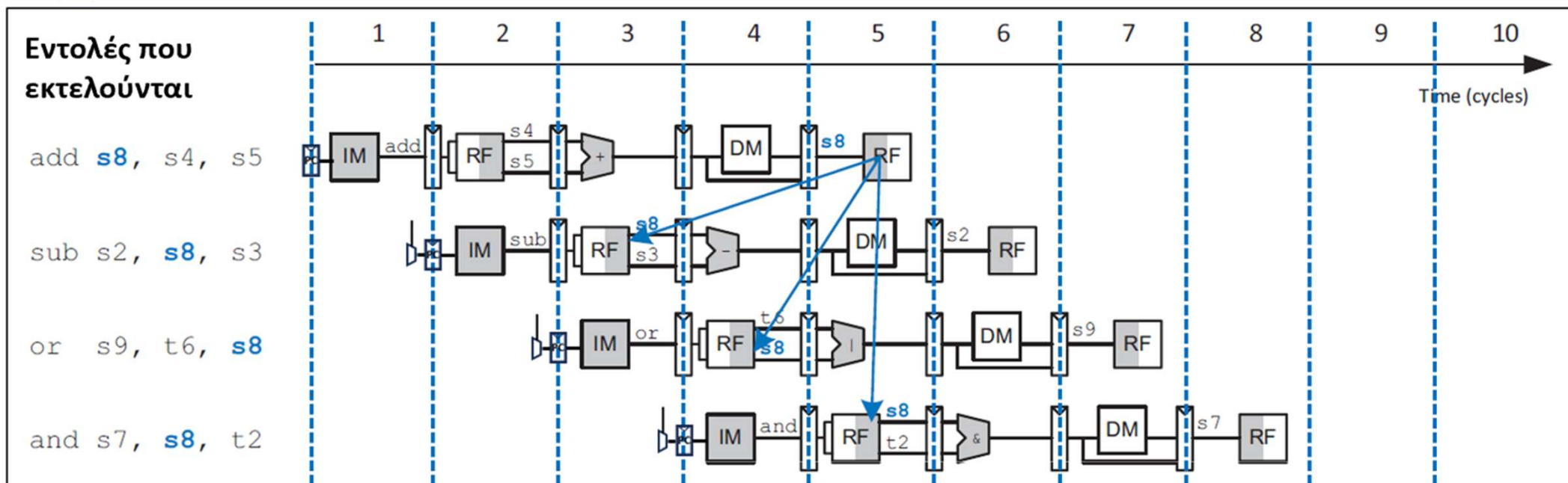
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι

- Οι επεξεργαστές πολλών κύκλων με υποστήριξη διοχέτευσης 5 σταδίων εκτελούν σε κάθε κύκλο ρολογιού το πολύ 5 εντολές ταυτόχρονα
 - μία σε κάθε διαφορετικό στάδιο, αρκεί να μην δημιουργούνται μεταξύ αυτών των εντολών εξαρτήσεις δεδομένων ή ελέγχου
- Όταν η εκτέλεση μιας εντολής εξαρτάται από τα αποτελέσματα άλλης εντολής που δεν έχει ολοκληρωθεί ακόμα, προκύπτει ένας **κίνδυνος (hazard)**
- Οι κίνδυνοι ταξινομούνται σε **κινδύνους δεδομένων (data hazards)** και **κινδύνους ελέγχου (control hazards)**
 - Κίνδυνος δεδομένων προκύπτει όταν μια εντολή προσπαθεί να διαβάσει δεδομένα από έναν καταχωρητή προορισμού προγενέστερης εντολής, τα οποία δεν έχουν ακόμα γραφεί ετεροχρονισμένα
 - Κίνδυνος ελέγχου προκύπτει όταν η απόφαση σχετικά με το ποια εντολή θα προσκομιστεί στη συνέχεια δεν έχει ληφθεί μέχρι τη στιγμή που λαμβάνει όντως χώρα η προσκόμιση της εντολής

Σε κάθε περίπτωση, πρέπει να αποσοβηθούν οι κίνδυνοι, ώστε το πρόγραμμα να εκτελείται σωστά παρά τη διοχέτευση

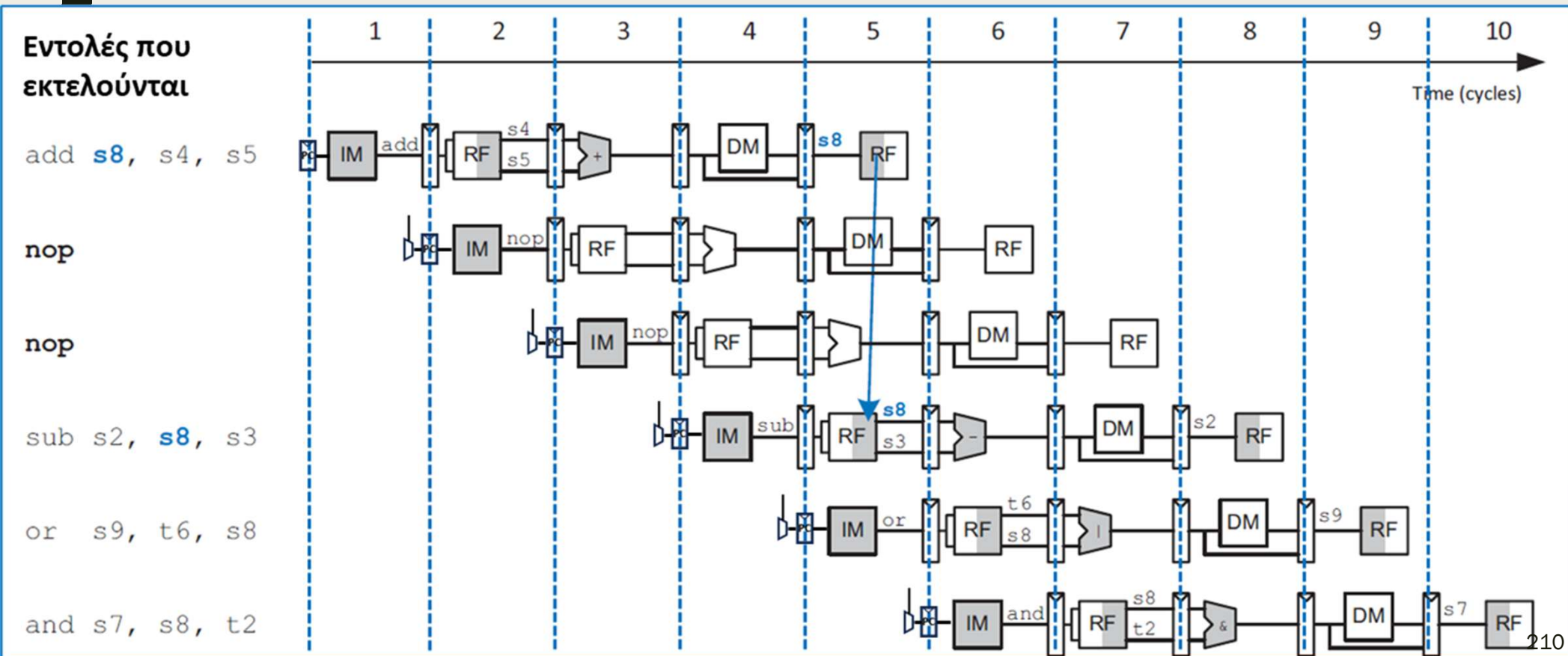
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Το αρχείο καταχωρητών εγγράφεται κατά τη διάρκεια του πρώτου μισού του κύκλου και διαβάζεται κατά τη διάρκεια του δεύτερου μισού του κύκλου
 - Δεν προκαλείται **κίνδυνος ανάγνωσης μετά από εγγραφή** (read after write, RAW), αν ο ίδιος καταχωρητής προορισμού (π.χ. s8), διαβάζεται στον ίδιο ή μεταγενέστερο κύκλο σε σχέση με τον κύκλο εγγραφής του (δηλαδή στον 5^ο κύκλο ή αργότερα)
 - Ο κίνδυνος εμφανίζεται όταν ο καταχωρητής προορισμού (π.χ. s8) διαβάζεται σε προηγούμενο κύκλο, δηλαδή όταν χρησιμοποιείται ως καταχωρητής προέλευσης στις δύο αμέσως επόμενες εντολές (SUB και OR)



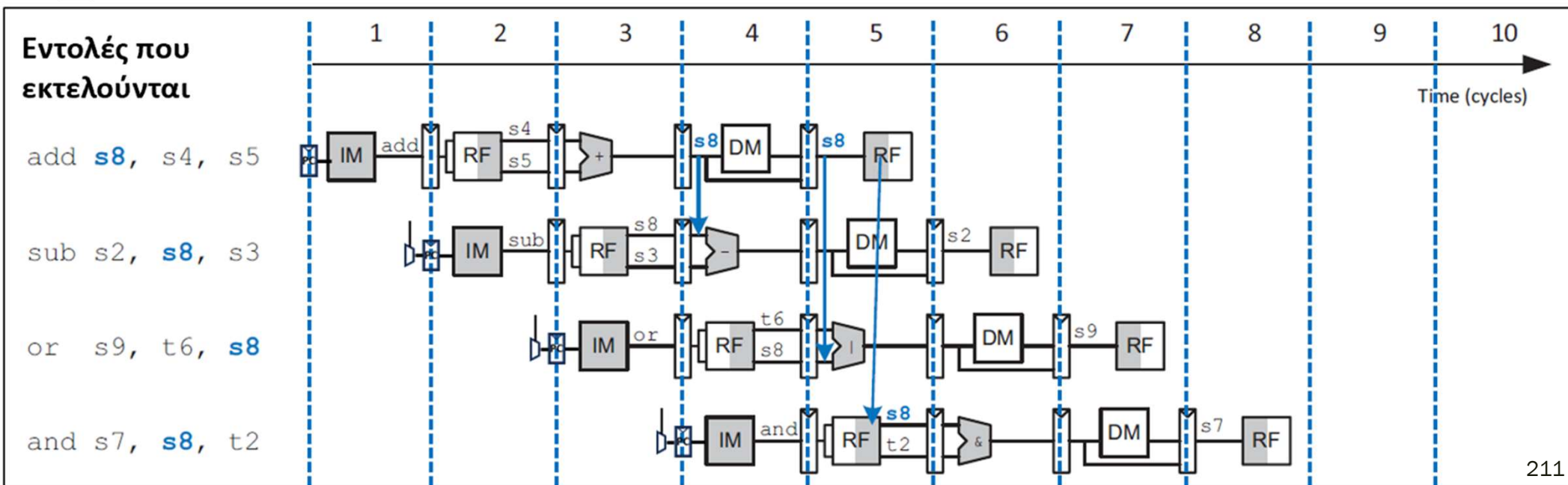
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Λύση βασισμένη στο λογισμικό για κίνδυνο δεδομένων τύπου RAW
 - Να εισαχθούν δύο εντολές NOP μεταξύ των εντολών AND και SUB, ώστε η εξαρτημένη εντολή να μην διαβάσει το αποτέλεσμα (s8) μέχρι αυτό να είναι διαθέσιμο στο αρχείο καταχωρητών
 - Μια τέτοια αλληλεμπλοκή λογισμικού (software interlock) όχι μόνο περιπλέκει τον προγραμματισμό, αλλά μειώνει και τις επιδόσεις, οπότε δεν είναι ιδανική επιλογή



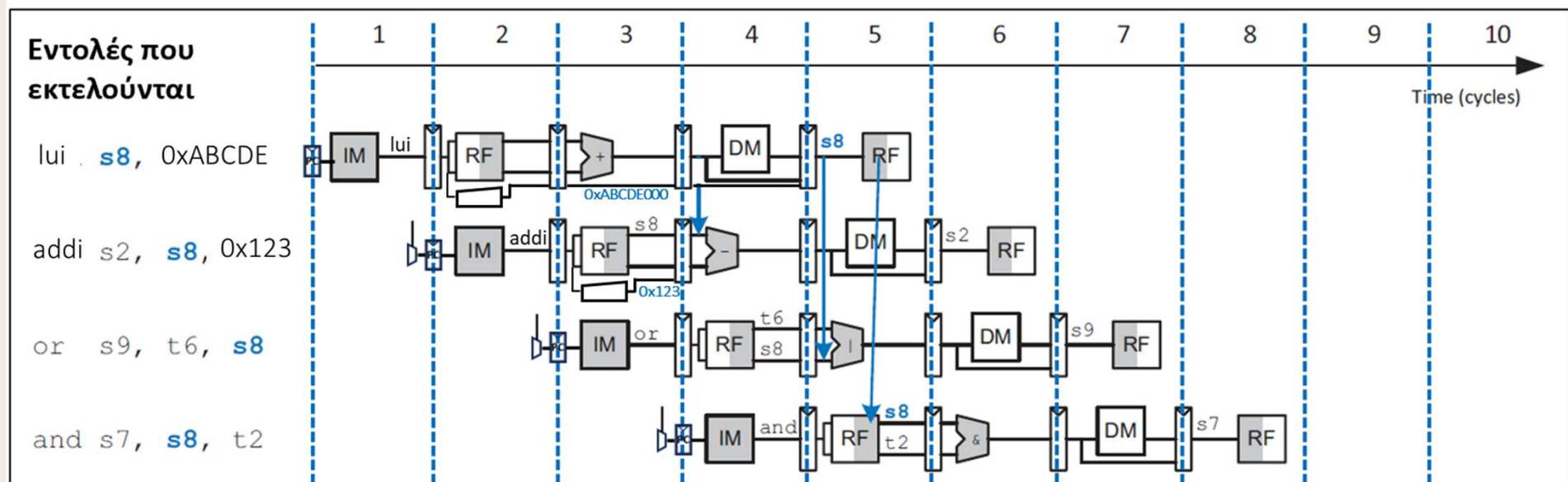
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
 - Το αποτέλεσμα πράξης στην ALU (ALUResult-3), που παράγεται στο στάδιο 3.EXECUTE, είναι διαθέσιμο και στο στάδιο 4.MEMORY (ALUResult-4) και στο στάδιο 5.WRITEBACK (Result-5) πριν την έγγραφή του στο αρχείο καταχωρητών (για εντολές ALU/ALUI)
 - Για παράδειγμα το s8 που παράγεται στον 3^ο κύκλο της ADD είναι διαθέσιμο:
 - στο στάδιο 4.MEMORY (ALUResult-4) (κύκλος 4) για να χρησιμοποιηθεί από την αμέσως επόμενη εντολή SUB, που βρίσκεται η εκτέλεσή της στο στάδιο 3.EXECUTE (κύκλος 4)
 - στο στάδιο 5.WRITEBACK πριν την εγγραφή στο RF (Result-5) (κύκλος 5) για να χρησιμοποιηθεί από την μεθεπόμενη εντολή OR, που βρίσκεται η εκτέλεσή της στο στάδιο 3.EXECUTE (κύκλος 5)
 - στο στάδιο 5.WRITEBACK μετά την εγγραφή στο RF (κύκλος 5) για να χρησιμοποιηθεί από την τρίτη κατά σειρά εντολή AND, που βρίσκεται η εκτέλεσή της στο στάδιο 2.DECODE (κύκλος 5) για άμεσο διάβασμα



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
Η ειδική περίπτωση του ζεύγους εντολών LUI-ADDI* για άμεσο τελεστέο των 32 bit
 - Ο άμεσος τελεστέος (Extimm-2) που παράγεται στο στάδιο 2.DECODE είναι διαθέσιμος και στο στάδιο 4.MEMORY (Extimm-4) και στο στάδιο 5.WRITEBACK (Result-5) πριν την έγγραφή του στο αρχείο καταχωρητών (για την εντολή LUI)
 - Για παράδειγμα το 0xABCDE000 που παράγεται στον 2^ο κύκλο της LUI είναι διαθέσιμο:
 - στο στάδιο 4.MEMORY (Extimm-4) (κύκλος 4) για να χρησιμοποιηθεί από την αμέσως επόμενη εντολή ADDI, που βρίσκεται η εκτέλεσή της στο στάδιο 3.EXECUTE (κύκλος 4)



* Εκτός από τις εντολές ALU/ALUI, άλλες εντολές με Rd, όπου το σήμα ελέγχου RegWrite-4 = 1, είναι οι LW, JALR και LUI. Η JALR αλλάζει την ροή του προγράμματος με κλήση διαδικασίας/συνάρτησης και αποθηκεύει το PC+4 στον ra. Η εντολή επιστροφής JR ra δεν είναι η αμέσως επόμενη εντολή που εκτελείται. Από την μεθεπόμενη εντολή το ra είναι διαθέσιμο είτε πριν είτε μετά την έγγραφή του στο RF. Η εντολή LW είναι η μόνη που απαιτεί παρακώλυση και θα μελετηθεί στην συνέχεια.

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
 - Τροποποίηση του επεξεργαστή με διοχέτευση με την προσθήκη δύο πολυπλεκτών προώθησης στο στάδιο 3.EXECUTE, ώστε να είναι άμεσα διαθέσιμα
 - το ALUResult-4 του σταδίου 4.MEMORY (για εντολές ALU/ALUI με Rd)
 - Το Result-5 (ALUResult-5) του σταδίου 5.WRITEBACK (για όλες τις εντολές με Rd)
 - το Extimm-4 του σταδίου 4.MEMORY (για το ζεύγος LUI-ADDI, αποκλειστικά στο SrcA)στις εισόδους SrcA-3 και SrcB-3 της μονάδας ALU
 - Προσθήκη μίας μονάδας διαχείρισης κινδύνων (Hazard Unit)
 - με εισόδους Rs1-3, Rs2-3, Rd-4, Rd-5, RegWrite-4, RegWrite-5 και ResultSrc(1:0)-4
 - με εξόδους, τα σήματα επιλογής των πολυπλεκτών προώθησης FrwdA(1:0)-3, FrwdB(1:0)-3
 - Οι πολυπλέκτες προώθησης λειτουργούν ως εξής:
 - SrcA = RD1-3, if FrwdA(1:0)-3 = “00” (χωρίς προώθηση)
 - SrcA = Extmimm-4, if FrwdA(1:0)-3 = “01” (προώθηση για το ζεύγος εντολών LUI-ADDI)
 - SrcA = ALUResult-4, if FrwdA(1:0)-3 = “10” (προώθηση για εντολές ALU/ALUI με Rd)
 - SrcA = Result-5, if FrwdA(1:0)-3 = “11” (προώθηση για για όλες τις εντολές με Rd)
 - SrcB = RD2-3, if FrwdB(1:0)-3 = “00” (χωρίς προώθηση)
 - SrcB = ALUResult-4, if FrwdB(1:0)-3 = “10” (προώθηση για εντολές ALU/ALUI με Rd)
 - SrcB = Result-5, if FrwdB(1:0)-3 = “11” (προώθηση για για όλες τις εντολές με Rd)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
 - Συνάρτηση προώθησης για αποφυγή κινδύνου δεδομένων RAW
 - $\text{SrcA-3} = \text{ALUResult-4}$ if $\text{Rs1-3} = \text{Rd-4}$, $\text{Rs1-3} \neq 0$, $\text{RegWrite-4} = '1'$, $\text{ResultSrc(1:0)-4} = "00"$
 - $\text{SrcA-3} = \text{Extimm-4}$ if $\text{Rs1-3} = \text{Rd-4}$, $\text{Rs1-3} \neq 0$, $\text{RegWrite-4} = '1'$, $\text{ResultSrc(1:0)-4} = "10"$
 - $\text{SrcA-3} = \text{Result-5}$ if $\text{Rs1-3} = \text{Rd-5}$, $\text{Rs1-3} \neq 0$, $\text{RegWrite-5} = '1'$
 - $\text{SrcB-3} = \text{ALUResult-4}$ if $\text{Rs2-3} = \text{Rd-4}$, $\text{Rs1-3} \neq 0$, $\text{RegWrite-4} = '1'$, $\text{ResultSrc(1:0)-4} = "00"$
 - $\text{SrcB-3} = \text{Result-5}$ if $\text{Rs2-3} = \text{Rd-5}$, $\text{Rs1-3} \neq 0$, $\text{RegWrite-5} = '1'$

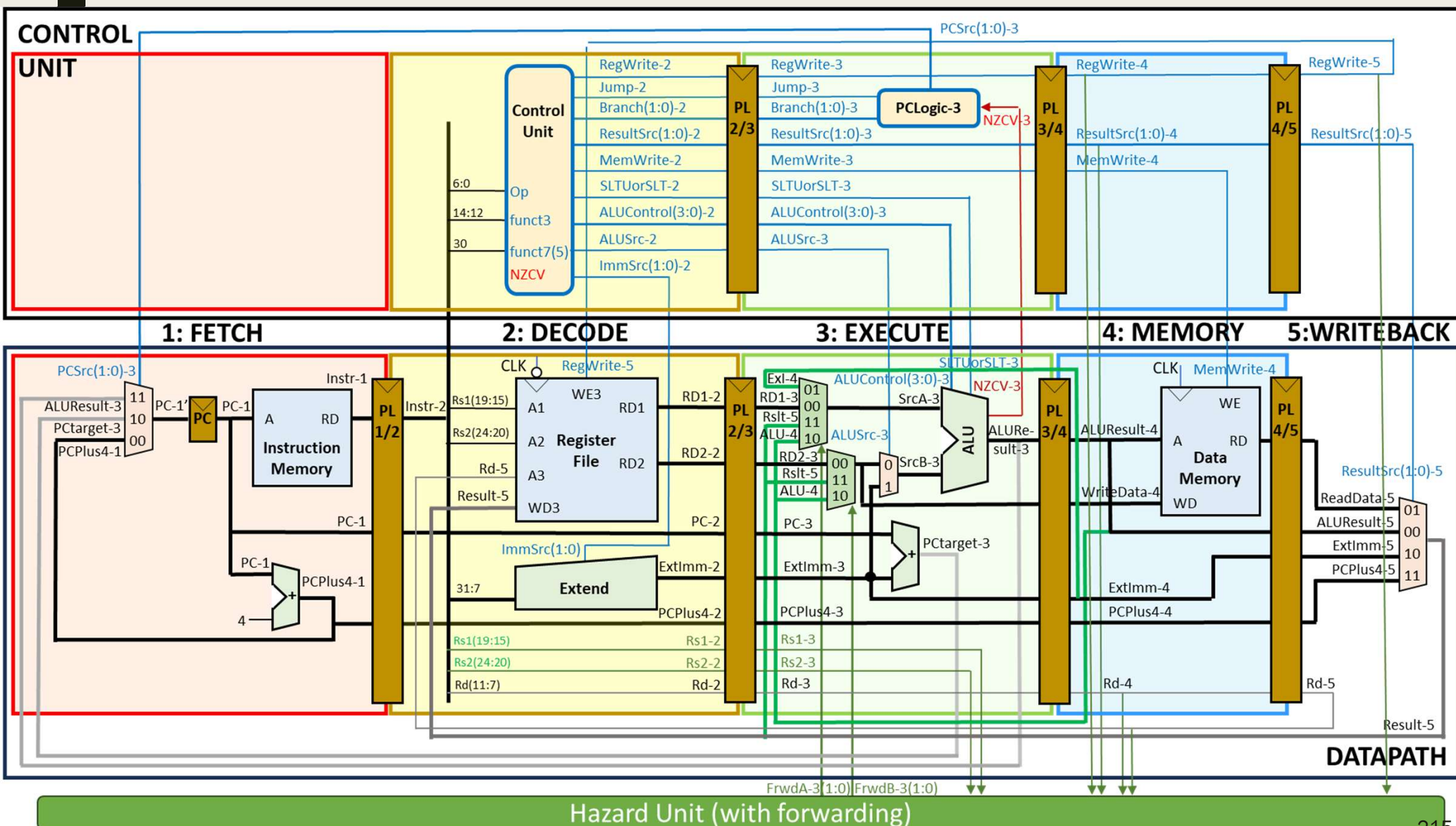
```
IF ((Rs1-3 = Rd-4) and (Rs1-3 /= "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "00"))
    THEN FrwdA(1:0)-3 = "10" - Forwarding SrcA-3 = ALUResult-4 (εντολές ALU/ALUI με Rd)
ELSIF ((Rs1-3 = Rd-4) and (Rs1-3 /= "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "10"))
    THEN FrwdA(1:0)-3 = "01" - Forwarding SrcA-3 = ExtImm-4 (LUI-ADDI)
ELSIF ((Rs1-3 = Rd-5) and (Rs1-3 /= "00000") and (RegWrite-5 = '1'))
    THEN FrwdA-3 = "11" - Forwarding SrcA-3 = Result-5 (όλες οι εντολές με Rd)
ELSE FrwdA-3 = "00" - No Forwarding (SrcA-3 = RD1-3)
```

```
IF ((Rs2-3 = Rd-4) and (Rs2-3 /= "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "00"))
    THEN FrwdB(1:0)-3 = "10" - Forwarding SrcB-3 = ALUResult-4 (εντολές ALU/ALUI με Rd)
ELSIF ((Rs2-3 = Rd-5) and (Rs2-3 /= "00000") and (RegWrite-5 = '1'))
    THEN FrwdB-3 = "11" - Forwarding SrcB-3 = Result-5 (όλες οι εντολές με Rd)
ELSE FrwdB-3 = "00" - No Forwarding (SrcB-3 = RD2-3)
```

Τα $\text{RegWrite-4} = 1$ και $\text{ResultSrc(1:0)-4} = 01$ προσδιορίζουν τις ALU/ALUI στο στάδιο 4.Memory.
Τα $\text{RegWrite-4} = 1$ και $\text{ResultSrc(1:0)-4} = 10$ προσδιορίζουν την LUI στο στάδιο 4.Memory.

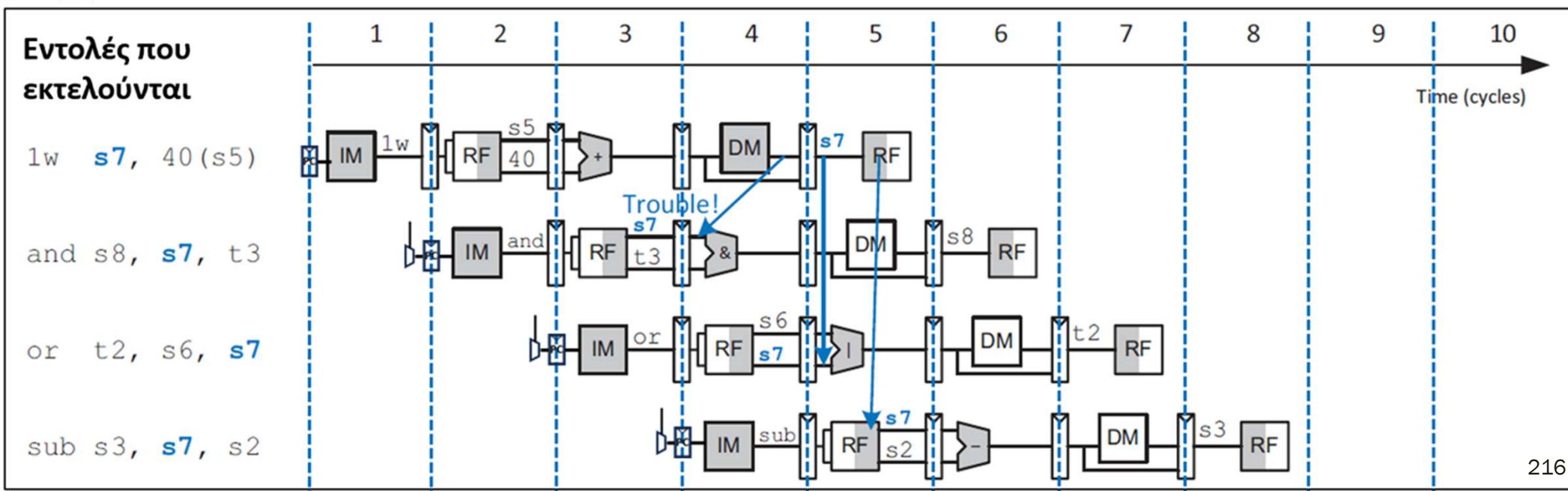
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Επεξεργαστής πολλών κύκλων με διοχέτευση και υποστήριξη προώθησης



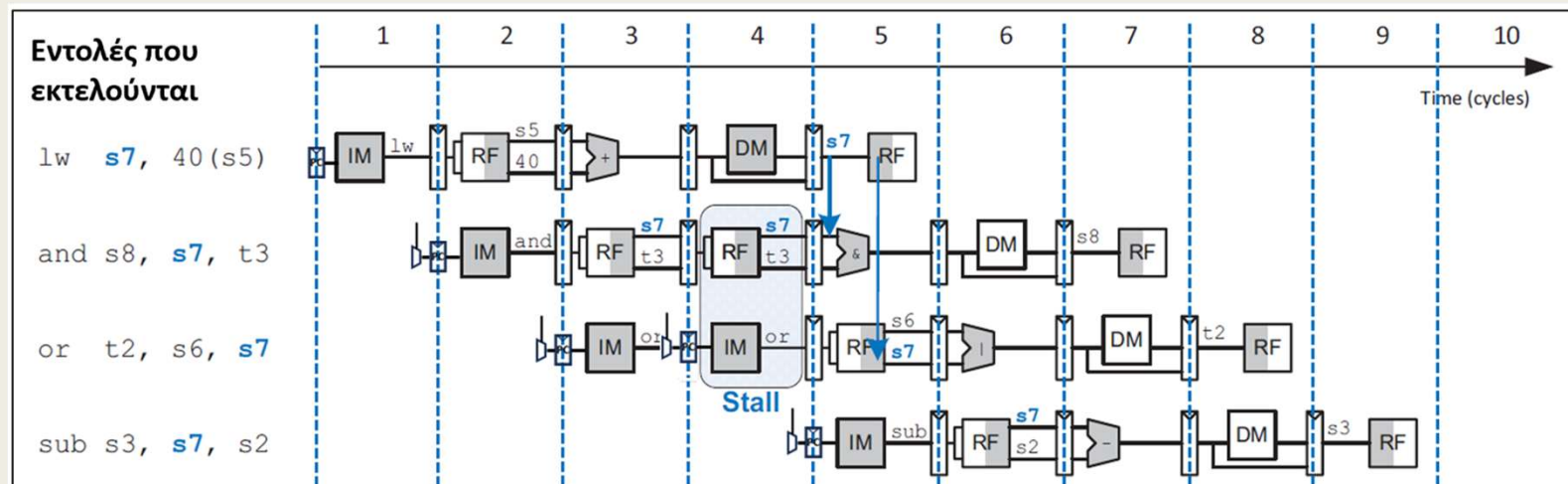
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
Απαίτηση για **παρακώλυση (stall)** της διοχέτευσης για την εντολή LW*
 - Το αποτέλεσμα της ανάγνωσης από την Μνήμη Δεδομένων που παράγεται στο στάδιο 4.MEMORY (ReadData-4) είναι διαθέσιμο μόνο στο στάδιο 5.WRITEBACK (Result-5) πριν την έγγραφή του στο αρχείο καταχωρητών
 - Για παράδειγμα το s7 που παράγεται στον 4^ο κύκλο της LW είναι διαθέσιμο:
 - στο στάδιο 4.MEMORY μετά το διάβασμα της μνήμης δεδομένων (ReadData-4) (στο τέλος του κύκλου 4) και **δεν δύναται να χρησιμοποιηθεί** από την αμέσως επόμενη εντολή AND, που βρίσκεται η εκτέλεσή της στο στάδιο 3.EXECUTE (στην αρχή του κύκλου 4)
 - Δεν επαρκεί η περίοδος του ρολογιού CLK



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της **προώθησης (forwarding)** για κίνδυνο δεδομένων τύπου RAW
Απαιτήση για **παρακώλυση (stall)** της διοχέτευσης για την εντολή LW
 - Η παρακώλυση κατά ένα κύκλο εφαρμόζεται στο στάδιο 2.DECODE της αμέσως επόμενης εντολής AND και στο στάδιο 1.FETCH της μεθεπόμενης εντολής OR (κύκλος 3)
 - η εντολή AND εισέρχεται στο στάδιο 2.Decode στον 3^ο κύκλο και παραμένει σε αυτό το στάδιο για ένα επιπλέον κύκλο (τον κύκλο 4)
 - η εντολή OR εισέρχεται στο στάδιο 1.FETCH στον 3^ο κύκλο και παραμένει σε αυτό το στάδιο για ένα επιπλέον κύκλο (τον κύκλο 4)
 - Για παράδειγμα το s7 που παράγεται στο τέλος του 4^{ου} κύκλου της LW είναι διαθέσιμο:
 - στο στάδιο 5.WRITEBACK πριν την εγγραφή στο RF (Result-5) (κύκλος 5) για να χρησιμοποιηθεί από την επόμενη εντολή AND, που βρίσκεται η εκτέλεσή της στο στάδιο 3.EXECUTE (κύκλος 5)
 - στο στάδιο 5.WRITEBACK μετά την εγγραφή στο RF (κύκλος 5) για να χρησιμοποιηθεί από την μεθεπόμενη εντολή OR, που βρίσκεται η εκτέλεσή της στο στάδιο 2.DECODE (κύκλος 5) για άμεσο διάβασμα από το RF



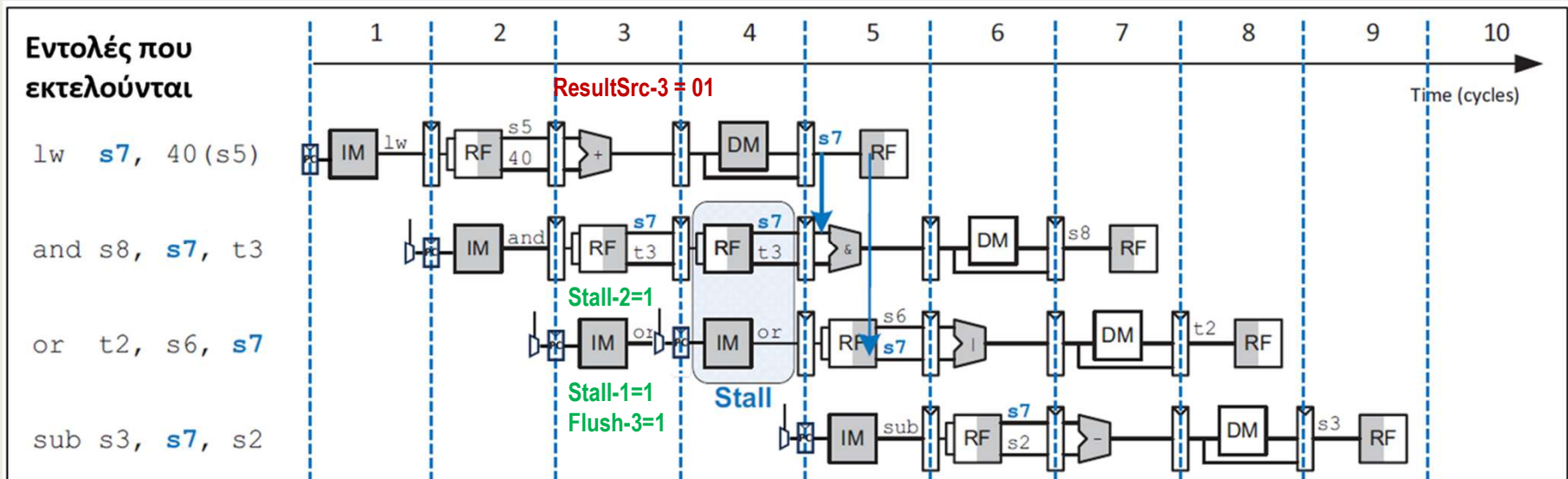
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Υλοποίηση της παρακώλυσης (stall) της διοχέτευσης για την εντολή LW
 - Η παρακώλυση του σταδίου 2.DECODE της επόμενης εντολής επιτυγχάνεται με την απενεργοποίηση εγγραφής του καταχωρητή διοχέτευσης PL1/2 (κύκλος 3)
 - Προσθήκη σήματος έγκρισης εγγραφής (WE) στον καταχωρητή PL1/2
 - Προσθήκη του σήματος Stall-2 στην μονάδα διαχείρισης κινδύνων (Hazard Unit)
 - Όταν παρακωλύεται ένα στάδιο, πρέπει να παρακωλυθούν στον ίδιο κύκλο και όλα τα προηγούμενα στάδια για να μην χαθεί καμία από τις επόμενες εντολές.
 - Η παρακώλυση του σταδίου 1.FETCH της μεθεπόμενης εντολής επιτυγχάνεται με την απενεργοποίηση εγγραφής του μετρητή προγράμματος PC (κύκλος 3)
 - Προσθήκη σήματος έγκρισης εγγραφής (WE) στον μετρητή προγράμματος PC
 - Προσθήκη του σήματος Stall-1 στην μονάδα διαχείρισης κινδύνων (Hazard Unit)
 - Ο καταχωρητής διοχέτευσης αμέσως μετά από το παρακωλυόμενο στάδιο 2.DECODE πρέπει να αδειάσει (εκκαθαριστεί) στον ίδιο κύκλο, ώστε να αποτραπεί η διάδοση ψευδών πληροφοριών προς τα εμπρός.
 - Απαιτείται εκκαθάριση του καταχωρητή διοχέτευσης PL2/3 (κύκλος 3)
 - Προσθήκη σύγχρονου Reset στον καταχωρητή PL2/3
 - Προσθήκη του σήματος Flush-3 στην μονάδα διαχείρισης κινδύνων (Hazard Unit)
 - Αναγνώριση της εκτέλεσης μίας εντολής LW στο στάδιο 3.EXECUTE (κύκλος 3)
 - Η εντολή LW είναι η μόνη που παράγει το σήμα ελέγχου ResultSrc(1:0)-3 = 01

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

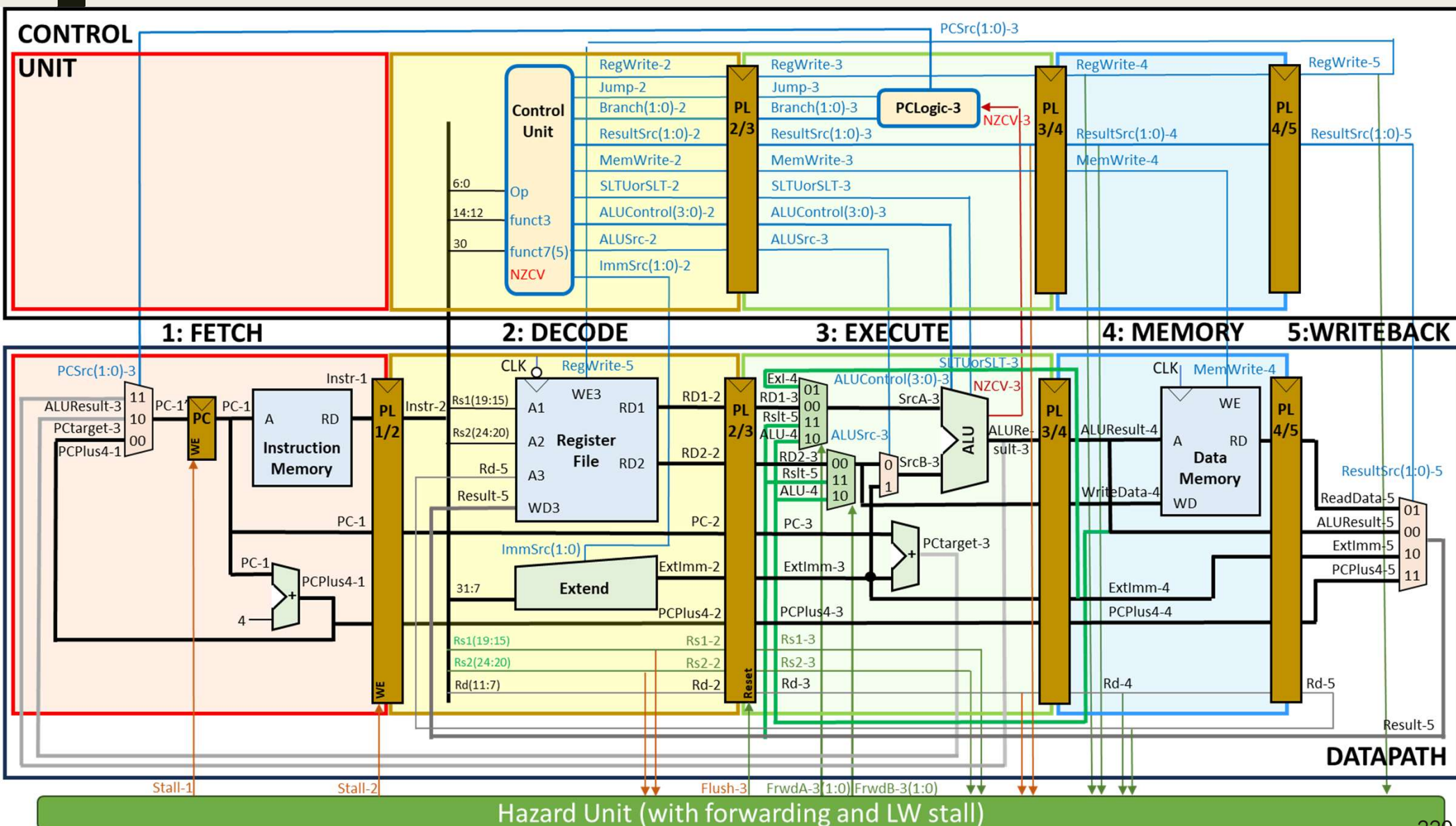
- Υλοποίηση της παρακώλυσης (stall) της διοχέτευσης για την εντολή LW
 - Συνάρτηση παρακώλυσης της διοχέτευσης για την εντολή LW στον 3^ο κύκλο
 - Η εντολή LW βρίσκεται στο στάδιο 3.EXECUTE και παράγει **ResultSrc(1:0)-3 = 01**
 - Η επόμενη εντολή AND βρίσκεται στο στάδιο 2.DECODE και πρέπει να παρακωλυθεί για έναν κύκλο στην περίπτωση που $Rs1-2 = Rd-3$ ή $Rs2-2 = Rd-3$
 - Το εσωτερικό σήμα **LWstall** της μονάδας διαχείρισης κινδύνων (Hazard Unit) ενεργοποιείται στον 3^ο κύκλο όταν ικανοποιείται η συνθήκη παρακώλυσης
 - ενεργοποιεί με την σειρά του τα σήματα **Stall-1**, **Stall-2** και **Flush-3** της μονάδας διαχείρισης κινδύνων (Hazard Unit)

IF ((ResultSrc(1:0)-3 = "01") and ((Rs1-2 = Rd-3) or (Rs2-2 = Rd-3)))
THEN LWstall = '1';
Stall-1 = LWstall; Stall-2 = LWstall; Flush-3 = LWstall;



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Επέκταση του επεξεργαστή με διοχέτευση για υποστήριξη παρακώλυσης της LW



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι ελέγχου

- **Κίνδυνος ελέγχου (Control Hazard)** προκύπτει όταν η απόφαση σχετικά με το ποια εντολή θα προσκομιστεί στη συνέχεια δεν έχει ληφθεί μέχρι τη στιγμή που λαμβάνει όντως χώρα η προσκόμιση της εντολής
 - *Επεξεργαστής με πρόβλεψη εκτέλεσης των 2 επόμενων κατά σειρά εντολών:*
Ανεξάρτητα εάν πρόκειται να εκτελεσθεί μία εντολή Branch ή JALR/JR στην διεύθυνση PC, για κάθε εντολή στη διεύθυνση PC προβλέπεται στους επόμενους δύο κύκλους να ξεκινήσει η εκτέλεση των δύο επόμενων κατά σειρά εντολών στις διευθύνσεις PC+4 και PC+8, αντίστοιχα
 - Όταν η εντολή στην διεύθυνση PC φθάσει στο στάδιο 3.Execute (3^{ος} κύκλος) παράγεται η πραγματική τιμή του σήματος **PCSrc(1:0)-3**
 - **PCSrc(1:0)-3 = "00"** (για όλες τις εντολές, εκτός Branch και JALR)
 - **PCSrc(1:0)-3 = "10"/"00"** (για τις Branch όταν ικανοποιείται/δεν ικανοποιείται η συνθήκη)
 - **PCSrc(1:0)-3 = "11"** (για τις εντολές JALR/JR)
 - Εάν **PCSrc(1)-3 = '0'** συνεχίζεται κανονικά η εκτέλεση του προγράμματος
 - Στον επόμενο 4^ο κύκλο γίνεται ανάκληση της εντολής στην διεύθυνση PC+12
 - Εάν **PCSrc(1)-3 = '1'** απαιτείται **απόρριψη αυτών των 2 εντολών (flush)**, που άρχισαν να εκτελούνται στις διευθύνσεις PC+4 και PC+8, αντίστοιχα, λόγω εσφαλμένης προβλέψεως και παρουσίας **κινδύνου ελέγχου (control hazard)**
 - Στον επόμενο 4^ο κύκλο γίνεται ανάκληση της εντολής είτε στην διεύθυνση **PCtarget-3** (για εντολές Branch που ικανοποιείται η συνθήκη (**PCSrc(1:0)-3 = "10"**), είτε στην διεύθυνση **ALUResult-3** για την εντολή JALR/JR (**PCSrc(1:0)-3 = "11"**)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι ελέγχου

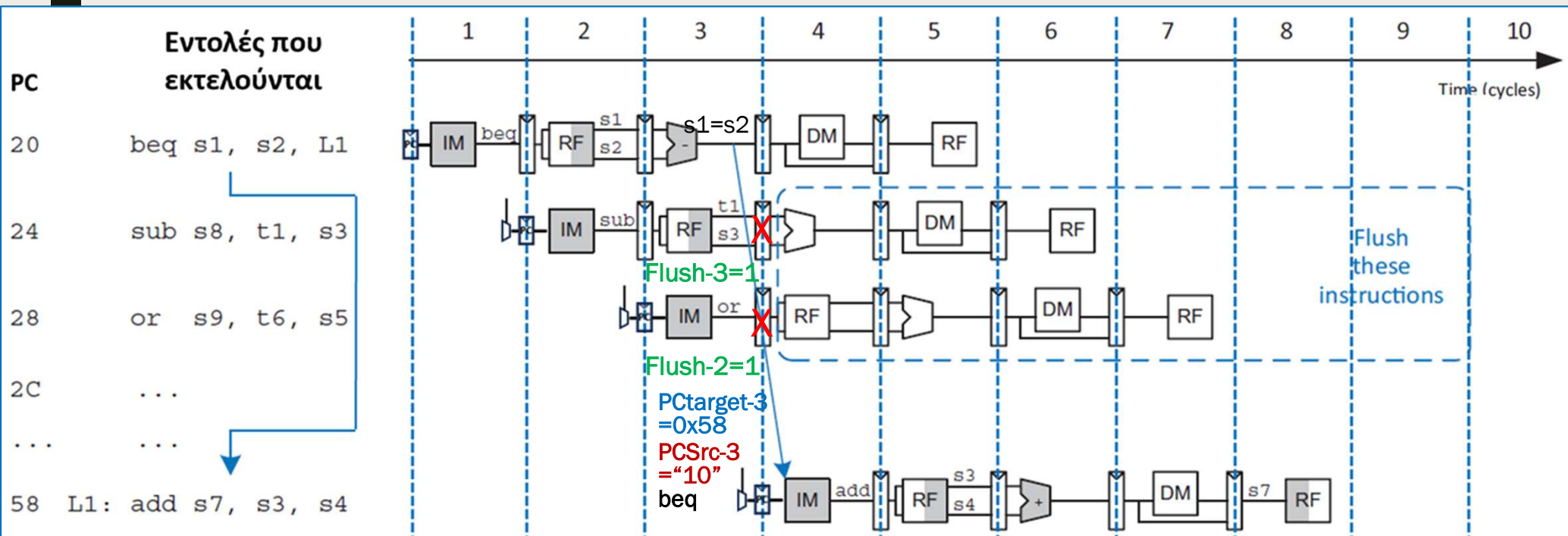
- Υλοποίηση της **απόρριψης (flush)** λόγω εσφαλμένης πρόβλεψης διακλάδωσης
 - Όταν στο στάδιο 3.EXECUTE μίας εντολής Branch ή JALR/JR παραχθεί το σήμα **PCSrc(1)-3 = '1'**, ο επεξεργαστής αναγνωρίζει την εσφαλμένη πρόβλεψη και απορρίπτει τις 2 εντολές που ήδη έχουν αρχίσει να εκτελούνται
 - Η απόρριψη επιτυγχάνεται μέσω του αδειάσματος των καταχωρητών διοχέτευσης PL1/2 και PL2/3
 - οι 2 περιττοί κύκλοι εντολών περιγράφονται με τον όρο ποινή εσφαλμένης πρόβλεψης διακλάδωσης (**branch misprediction penalty**).
 - Απαιτείται εκκαθάριση του καταχωρητή διοχέτευσης PL1/2 (κύκλος 3)
 - Προσθήκη σύγχρονου Reset στον καταχωρητή PL1/2
 - Προσθήκη του σήματος **Flush-2** στην μονάδα διαχείρισης κινδύνων (**Hazard Unit**)
 - Απαιτείται εκκαθάριση του καταχωρητή διοχέτευσης PL2/3 (κύκλος 3)
 - Προσθήκη σύγχρονου Reset στον καταχωρητή PL2/3
 - Προσθήκη του σήματος **Flush-3** στην μονάδα διαχείρισης κινδύνων (**Hazard Unit**)
 - Χρησιμοποιείται και στην παρακώλυση (stall) της διοχέτευσης για την εντολή LW
 - Λογική απόρριψης για την μονάδα διαχείρισης κινδύνων ελέγχου

Flush-2 = PCSrc(1)-3 = '1'

Flush-3 = PCSrc(1)-3 = '1' or LWstall

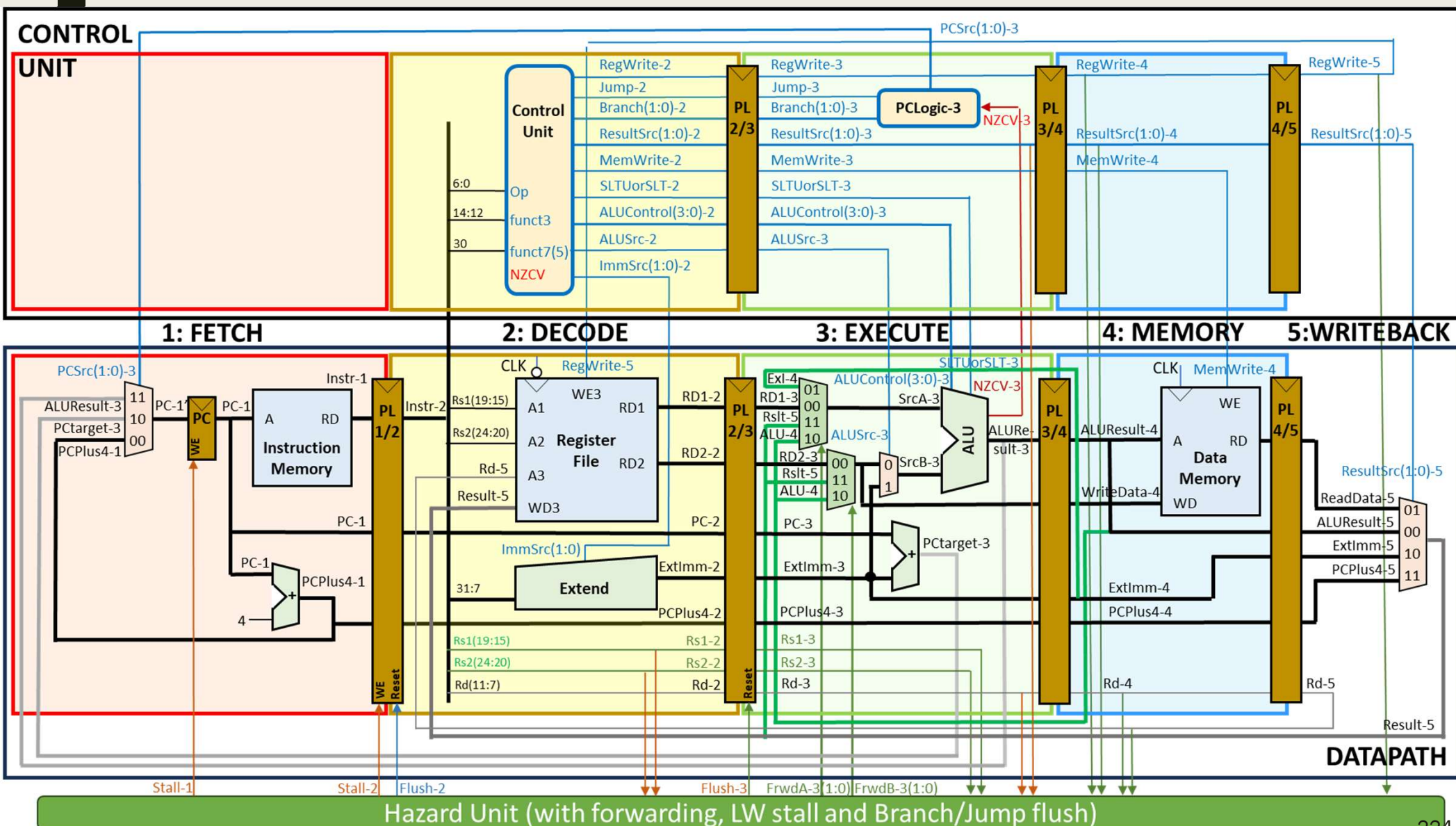
Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι ελέγχου

- Υλοποίηση της **απόρριψης (flush)** λόγω εσφαλμένης πρόβλεψης διακλάδωσης
 - Στο παράδειγμα φαίνεται μια περίπτωση απόρριψης στην οποία ακολουθείται μια διακλάδωση από την διεύθυνση $PC=0x20$ στην διεύθυνση $PCtarget=0x58$
 - Στο στάδιο 3.EXECUTE της εντολής BEQ ικανοποιείται η συνθήκη ($s1=s2$) και παράγεται το ορθό $PCSrc(0:1)-3 = \text{"10"}$. Παράλληλα υπολογίζεται το $PCtarget-3 = 0x58$
 - Στον ίδιο κύκλο η μονάδα διαχείρισης κινδύνων θέτει $Flush-2 = 1$ και $Flush-3 = 1$
 - Στον επόμενο 4^ο κύκλο γίνεται ανάκληση της εντολής στην διεύθυνση $0x58$ και εκκαθάριση του καταχωρητών διοχέτευσης PL1/2 και PL2/3



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι ελέγχου

- Επέκταση του επεξεργαστή με διοχέτευση για υποστήριξη απόρριψης



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα κινδύνων

- Σύνοψη της Μονάδας Διαχείρισης Κινδύνων (Hazard Unit)
 - Στους επεξεργαστές πολλών κύκλων με υποστήριξη διοχέτευσης διαχειρίζεται κινδύνους δεδομένων και ελέγχου
 - κίνδυνοι δεδομένων τύπου RAW προκύπτουν όταν μια εντολή εξαρτάται από αποτέλεσμα προηγούμενης εντολής το οποίο δεν έχει εγγραφεί ακόμα στο αρχείο καταχωρητών
 - αποσοβούνται με χρήση της **προώθησης (forwarding)** αν το αποτέλεσμα υπολογίζεται αρκετά νωρίς και της **παρακώλυσης (stall)** μέχρι το αποτέλεσμα να γίνει διαθέσιμο για προώθηση (στις εντολές LW)
 - Κίνδυνοι ελέγχου προκύπτουν όταν η απόφαση για την επόμενη εντολή προς προσκόμιση δεν έχει ληφθεί μέχρι τη στιγμή που αυτή πρέπει να προσκομιστεί
 - αποσοβούνται μέσω της πρόβλεψης εκτέλεσης των 2 επόμενων κατά σειρά εντολών και της εκκαθάρισης της διοχέτευσης, αν αργότερα η πρόβλεψη αποδειχθεί εσφαλμένη

Inputs	Outputs
Rs1-2, Rs1-3	Stall-1, Stall-2 (WE PC, PL1/2)
Rs2-2, Rs2-3	Flush-2, Flush-3 (reset PL1/2, PL2/3)
Rd-3, Rd-4, Rd-5	FrwdA-3(1:0), FrwdB-3(1:0)
ResultSrc(1:0)-3, ResultSrc(1:0)-4,	
RegWrite-4, RegWrite-5	

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): μονάδα κινδύνων

■ Σύνοψη της Μονάδας Διαχείρισης Κινδύνων (Hazard Unit)

- Λογική διαχείρισης κινδύνων δεδομένων τύπου RAW με προώθηση

```
IF ((Rs1-3 = Rd-4) and (Rs1-3 != "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "00"))  
  THEN FrwdA(1:0)-3 = "10" - Forwarding SrcA-3 = ALUResult-4 (εντολές ALU/ALUI με Rd)  
ELSIF ((Rs1-3 = Rd-4) and (Rs1-3 != "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "10"))  
  THEN FrwdA(1:0)-3 = "01" - Forwarding SrcA-3 = ExtImm-4 (LUI-ADDI)  
ELSIF ((Rs1-3 = Rd-5) and (Rs1-3 != "00000") and (RegWrite-5 = '1'))  
  THEN FrwdA-3 = "11" - Forwarding SrcA-3 = Result-5 (όλες οι εντολές με Rd)  
ELSE FrwdA-3 = "00" - No Forwarding (SrcA-3 = RD1-3)
```

```
IF ((Rs2-3 = Rd-4) and (Rs2-3 != "00000") and (RegWrite-4 = '1') and (ResultSrc(1:0)-4 = "00"))  
  THEN FrwdB(1:0)-3 = "10" - Forwarding SrcB-3 = ALUResult-4 (εντολές ALU/ALUI με Rd)  
ELSIF ((Rs2-3 = Rd-5) and (Rs2-3 != "00000") and (RegWrite-5 = '1'))  
  THEN FrwdB-3 = "11" - Forwarding SrcB-3 = Result-5 (όλες οι εντολές με Rd)  
ELSE FrwdB-3 = "00" - No Forwarding (SrcB-3 = RD2-3)
```

- Λογική παρακώλυσης (stall) της διοχέτευσης για την εντολή LW

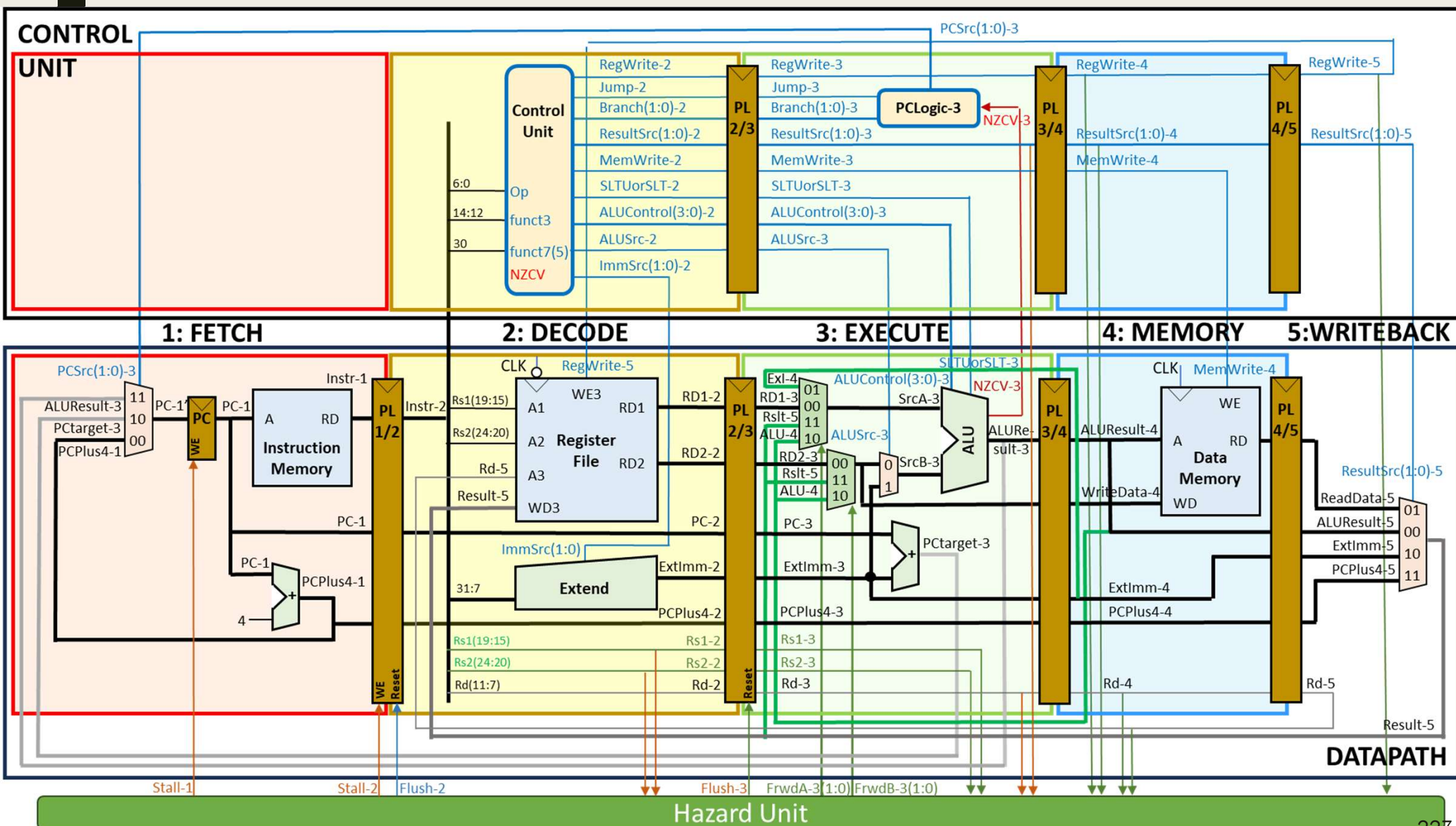
```
IF ((ResultSrc(1:0)-3 = "01") and ((Rs1-2 = Rd-3) or (Rs2-2 = Rd-3))) THEN LWstall = '1';  
Stall-1 = LWstall; Stall-2 = LWstall; Flush-3 = LWstall;
```

- Λογική απόρριψης 2 εντολών λόγω εσφαλμένης πρόβλεψης διακλάδωσης

```
Flush-2 = PCSrc(1)-3 = '1'  
Flush-3 = PCSrc(1)-3 = '1' or LWstall
```

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): κίνδυνοι δεδομένων

- Ολοκληρωμένος επεξεργαστής με διοχέτευση και διαχείριση κινδύνων δεδομένων και ελέγχου



Ανάλυση επιδόσεων επεξεργαστών

■ Πώς επηρεάζεται η επίδοση

- $\text{CPU}_{\text{time}} = (\# \text{ instructions/program}) \times \text{CPI} \times T_c = \text{seconds/program}$

■ Μέσος αριθμός κύκλων ανά εντολή (cycles per instruction - CPI):

- μέσος αριθμός κύκλων ανά εντολή του ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ενός προγράμματος
- εξαρτάται από το μίγμα των εντολών σε ένα πρόγραμμα (πόσες εντολές είναι επεξεργασίας δεδομένων, μνήμης ή διακλάδωσης)

■ Εντολές ανά κύκλο (instructions per cycle - IPC):

- διεκπεραιωτική ικανότητα - το αντίστροφο του CPI

■ Περίοδος T_c του ρολογιού (clock period):

- το πλήθος των νανοδευτερολέπτων ανά κύκλο ρολογιού
- καθορίζεται από την **κρίσιμη διαδρομή (critical path)** μέσω της λογικής της εκάστοτε μικροαρχιτεκτονικής του επεξεργαστή
- ο τρόπος σχεδίασης της διαδρομής δεδομένων και των μονάδων που την απαρτίζουν επηρεάζουν την **κρίσιμη διαδρομή (critical path)** και συνεπώς την περίοδο του ρολογιού

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

- Η τιμή του **CPI** στον επεξεργαστή με διοχέτευση εξαρτάται από:
 - την ποσόστωση (%) των πιθανών κατηγοριών εντολών που εκτελούνται σε ένα συγκεκριμένο πρόγραμμα
 - με την εφαρμογή της προώθησης, οι εντολές (εκτός JALR) έχουν $CPI = 1$, εάν δεν προκαλείται παρακώλυση ή απόρριψη. Η εντολή JALR/JR έχει $CPI = 3$
 - την ποσόστωση (40%) των πιθανών παρακωλύσεων (καθυστέρηση ενός κύκλου) κατά την εκτέλεση της εντολής LW ($CPI = 2$)
 - την ποσόστωση (50%) των πιθανών απορρίψεων λόγω εσφαλμένης πρόβλεψης διακλάδωσης (καθυστέρηση δύο κύκλων) στις εντολές Branch ($CPI = 3$)
- Παράδειγμα υπολογισμού CPI:

Κατηγορία εντολών	Ποσόστωση παρακώλ. / απορ.	Υπολογισμός CPI ανά κατηγορία εντολών	Ποσόστωση εντολών	Υπολογισμός CPI (τελικός)
ALU/ALUI	0%	$100\% \times 1 = 1$	50%	0,50
LW	40%	$60\% \times 1 + 40\% \times 2 = 1,4$	20%	0,28
SW	0%	$100\% \times 1 = 1$	10%	0,10
Branch	50%	$50\% \times 1 + 50\% \times 3 = 2$	6%	0,12
JALR/JR	0%	$100\% \times 3 = 3$	4%	0,12
LUI	0%	$100\% \times 1 = 1$	10%	0,10
			100%	1,22

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

- Η περίοδος του CLK καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του **πιο αργού σταδίου της εντολής** μέσω της λογικής της μικροαρχιτεκτονικής πολλών κύκλων με διοχέτευση
- Το αρχείο καταχωρητών χρησιμοποιείται δύο φορές σε κάθε κύκλο:
 - εγγράφεται στο πρώτο μισό του σταδίου 5.*Writeback*, και
 - διαβάζεται στο δεύτερο μισό του σταδίου 2.*Decode*.

Συνεπώς, η διπλάσια κρίσιμη διαδρομή για καθένα από αυτά τα στάδια πρέπει να χωράει σε έναν κύκλο
- Στο στάδιο 3.*EXECUTE* η κρίσιμη διαδρομή προκύπτει με βάση την αντίστοιχη διαδρομή του επεξεργαστή πολλών κύκλων με την προσθήκη της υλοποίησης της προώθησης του ALUResult-5 στο SrcB-3 της μονάδας ALU από το στάδιο 5.*WRITEBACK* πριν την εγγραφή στο RF (Result-5) κατά την εκτέλεση του 3^{ου} κύκλου των εντολών Branch
 - Υποθέτουμε ότι η καθυστέρηση για τον υπολογισμό των *FrwdA(1:0)-3* και *FrwdB(1:0)-3* της μονάδας διαχείρισης κινδύνων είναι μικρότερη από ή ίση με την καθυστέρηση του πολυπλέκτη 4 σε 1 με έξοδο Result-5 (30 ps)

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

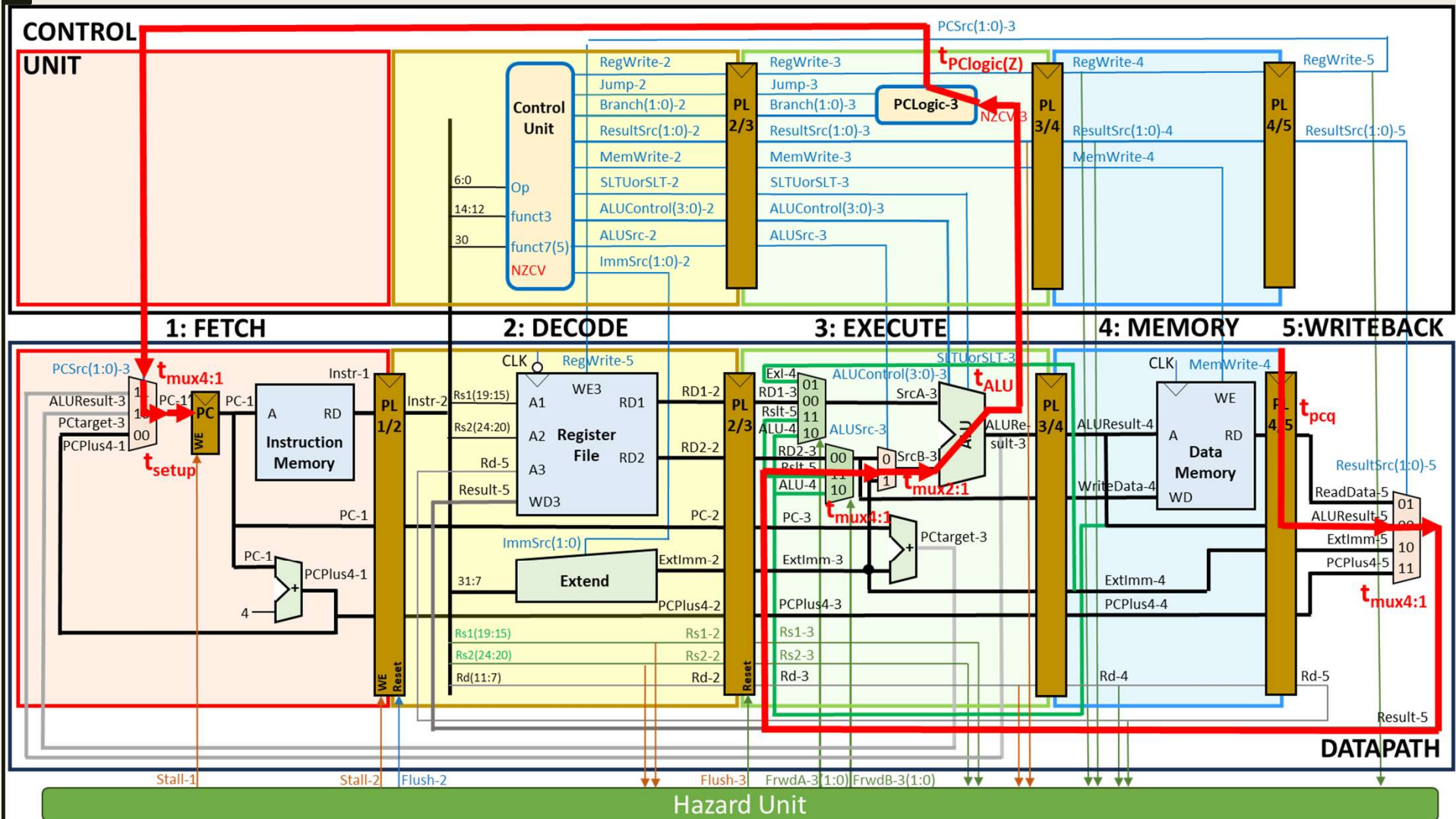
- Οι κρίσιμες διαδρομές ανά στάδιο υπολογίζονται ως ακολούθως:
 - 1.FETCH: $t_{pcq} + t_{IM_read} + t_{setup} = (40 + 200 + 50) \text{ ps} = 290 \text{ ps}$
 - 2.DECODE: $2 \times (t_{RF_read} + t_{setup}) = 2 \times (100 + 50) \text{ ps} = 300 \text{ ps}$
 - 3.EXECUTE: $t_{pcq} + 2 \times t_{mux4:1} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup} = (40 + 2 \times 30 + 20 + 150 + 10 + 30 + 50) \text{ ps} = 360 \text{ ps}$
 - 4.MEMORY: $t_{pcq} + t_{DM_read} + t_{setup} = (40 + 200 + 50) \text{ ps} = 290 \text{ ps}$
 - 5.WRITEBACK: $2 \times (t_{pcq} + t_{mux4:1} + t_{RF_setup}) = 2 \times (40 + 30 + 60) \text{ ps} = 260 \text{ ps}$
 - Η μεγαλύτερη από τις κρίσιμες διαδρομές εμφανίζεται στο στάδιο 3.EXECUTE και προσδιορίζει την περίοδο του ρολογιου CLK ($T_c = 360 \text{ ps}$)

Ενδεικτικές τιμές σε τεχνολογία CMOS 7-nm.

Στοιχεία	Παράμετρος	Καθυστέρηση (ps)
Καθυστέρηση από το CLK έως την έξοδο Q (καταχωρητής)	T_{pcq}	40
Χρόνος σταθεροποίησης καταχωρητή	T_{setup}	50
Χρόνος σταθεροποίησης αρχείου καταχωρητών	t_{RF_setup}	60
Καθ. διάδ. στην μονάδα ελέγχου InstrDec/ALUDec/PCLogic	$t_{IND}/t_{ALUD}/t_{PClogic}$	25/35/35(10 από NZCV)
Καθυστέρηση διάδοσης (control path) στη μονάδα Extend	t_{Ext}	35
Καθυστέρηση διάδοσης πολυπλέκτη 2σε1/4σε1/8σε1	$T_{mux2:1/4:1/8:1}$	20/30/50
Καθυστέρηση διάδοσης στη μονάδα ALU/στον αθροιστή	t_{ALU}/t_{adder}	150/100
Χρόνος ανάγνωσης από τη μνήμη εντολών (ROM)	t_{IM_read}	200
Χρόνος ανάγνωσης από τη μνήμη δεδομένων (RAM)	t_{DM_read}	200
Χρόνος ανάγνωσης από το αρχείο καταχωρητών (RAM)	t_{RF_read}	100

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

- Η εκτέλεση του 3^{ου} κύκλου των εντολών Branch με προώθηση στο SrcB-3
 - $T_c = t_{pcq} + 2 \times t_{mux4:1} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup} =$
 $(40 + 2 \times 30 + 20 + 150 + 10 + 30 + 50) \text{ ps} = 360 \text{ ps}$



Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

■ Επεξεργαστής ενός κύκλου (CPI = 1)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση της **εντολής LW**

- $T_c = t_{pcq} + t_{IM_read} + t_{RF_read} + t_{ALU} + t_{DM_read} + t_{mux4:1} + t_{RF_setup}$

- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:

- $T_c = (40 + 200 + 100 + 150 + 200 + 30 + 60) \text{ ps} = 780 \text{ ps}$

- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:

- $\text{CPU}_{\text{time}} = (\# \text{ instr/program}) \times \text{CPI} \times T_c = 100 \times 10^9 \times 1 \times 780 \times 10^{-12} = 78 \text{ s}$

■ Επεξεργαστής πολλών κύκλων (CPI = 4)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του 3^{ου} κύκλου των εντολών **Branch**

- $T_c = t_{pcq} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup}$

- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:

- $T_c = (40 + 20 + 150 + 10 + 30 + 50) \text{ ps} = 300 \text{ ps}$

- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:

- $\text{CPU}_{\text{time}} = (\# \text{ instr/program}) \times \text{CPI} \times T_c = 100 \times 10^9 \times 4 \times 300 \times 10^{-12} = 120 \text{ s}$

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

■ Επεξεργαστής πολλών κύκλων (CPI = 4)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του 3^{ου} κύκλου των εντολών Branch

$$\blacksquare T_c = t_{pcq} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup}$$

- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:

$$\blacksquare T_c = 40 \text{ ps} + 20 \text{ ps} + 150 \text{ ps} + 10 \text{ ps} + 30 \text{ ps} + 50 \text{ ps} = 300 \text{ ps}$$

- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:

$$\blacksquare CPU_{time} = (\# \text{ instr/program}) \times CPI \times T_c = 100 \times 10^9 \times 4 \times 300 \times 10^{-12} = 120 \text{ s}$$

■ Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (CPI = 1,22)

- Η περίοδος του CLK (T_c) καθορίζεται από την **κρίσιμη διαδρομή (critical path)** που ενεργοποιείται κατά την εκτέλεση του 3^{ου} κύκλου των εντολών Branch με την προσθήκη της προώθησης του ALUResult-5 στο SrcB-3 της μονάδας ALU

$$\blacksquare T_c = t_{pcq} + 2 \times t_{mux4:1} + t_{mux2:1} + t_{ALU} + t_{PClogic(Z)} + t_{mux4:1} + t_{setup}$$

- η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:

$$\blacksquare T_c = (40 + 2 \times 30 + 20 + 150 + 10 + 30 + 50) \text{ ps} = 360 \text{ ps}$$

- ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:

$$\blacksquare CPU_{time} = (\# \text{ instr/program}) \times CPI \times T_c = 100 \times 10^9 \times 1,22 \times 360 \times 10^{-12} = 44 \text{ s}$$

Επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης (pipelining): ανάλυση επιδόσεων

- Ο επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης είναι σημαντικά **πιο γρήγορος από τους άλλους δύο επεξεργαστές**
 - Ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι **44 s** για τον επεξεργαστή με διοχέτευση, συγκρινόμενος με τα **78 s** για τον επεξεργαστή ενός κύκλου και τα **120 s** για τον επεξεργαστή πολλών κύκλων (χωρίς διοχέτευση)
 - Το πλεονέκτημά του έναντι του επεξεργαστή ενός κύκλου σίγουρα δεν πλησιάζει την πενταπλάσια επιτάχυνση που θα ήλπιζε κανείς να πετύχει με μια διοχέτευση πέντε σταδίων λόγω:
 - των κινδύνων δεδομένων και ελέγχου που αυξάνουν ελαφρώς το CPI (π.χ. από 1 σε 1,22)
 - της επιβάρυνσης δημιουργίας ακολουθίας των καταχωρητών διοχέτευσης που προκαλείται σε κάθε στάδιο της διοχέτευσης, και όχι μόνο μία φορά για τη συνολική διαδρομή δεδομένων
 - την μη εξισορροπημένη καθυστέρηση διάδοσης στα στάδια της διοχέτευσης
- Ο επεξεργαστής πολλών κύκλων με υποστήριξη διοχέτευσης μπορεί να γίνει ακόμα πιο γρήγορος αν τα στάδια της διοχέτευσης γίνουν πιο εξισορροπημένα
 - Η επίλυση της διακλάδωσης στο στάδιο 3.Execute απαιτεί περισσότερο χρόνο από οποιοδήποτε άλλο στάδιο. Μία βελτίωση θα ήταν να συμπεριλάβουμε τον πολυπλέκτη Result στο στάδιο 4.Memory, μειώνοντας την περίοδο του CLK (T_c) στα 330 ps

Επεξεργαστές πολλών κύκλων είχαν τα υπολογιστικά συστήματα IBM 360, DEC VAX και οι μικρο-επεξεργαστές 6502 και 8088. Υπάρχουν ακόμα ως φθινοί μικροελεγκτές για οικιακές συσκευές και παιχνίδια. Αντικαθίστανται από επεξεργαστές με διοχέτευση. Η Intel εισήγαγε την διοχέτευση από τον 80486 το 1989. Όλοι οι RISC υποστηρίζουν σήμερα διοχέτευση.

Συγκρίσεις μεταξύ των τριών βασικών μικροαρχιτεκτονικών RISC-V

■ Σύγκριση απόδοσης

Μικροαρχιτεκτονική	Ενός κύκλου	Πολλών κύκλων	Με διοχέτευση
Μέγιστη καθυστέρηση διάδοσης	780 ps	300 ps	360 ps
CPI	1	4	1,22
Διεκπεραιωτική ικανότητα (Throughput)	$1/780\text{ps} = 1,28 \text{ Ginst/s}$	$1/(4 \times 300)\text{ps} = 0,83 \text{ Ginst/s}$	$1/(1,22 \times 360)\text{ps} = 2,27 \text{ Ginst/s}$
Χρόνος εκτέλεσης προγρ. 100 δις εντολών	78 s	120 s	44 s
Λανθάνων χρόνος (Latency)	780 ps	$5 \times 300 \text{ ps} = 1.500 \text{ ps}$	$5 \times 360 \text{ ps} = 1.800 \text{ ps}$

■ Σύγκριση HW

Μικροαρχιτεκτονική	Ενός κύκλου	Πολλών κύκλων	Με διοχέτευση
Βασικές μονάδες	ALU, RegisterFile, Extend, 2x Adders, Control, InstrMemory, DataMemory		
Καταχωρητές 32 bit	1	7	19
Πολυπλέκτες	3	3	5
Επιπλέον λογική			Hazard Unit