Έκδοση RISC_V 2025

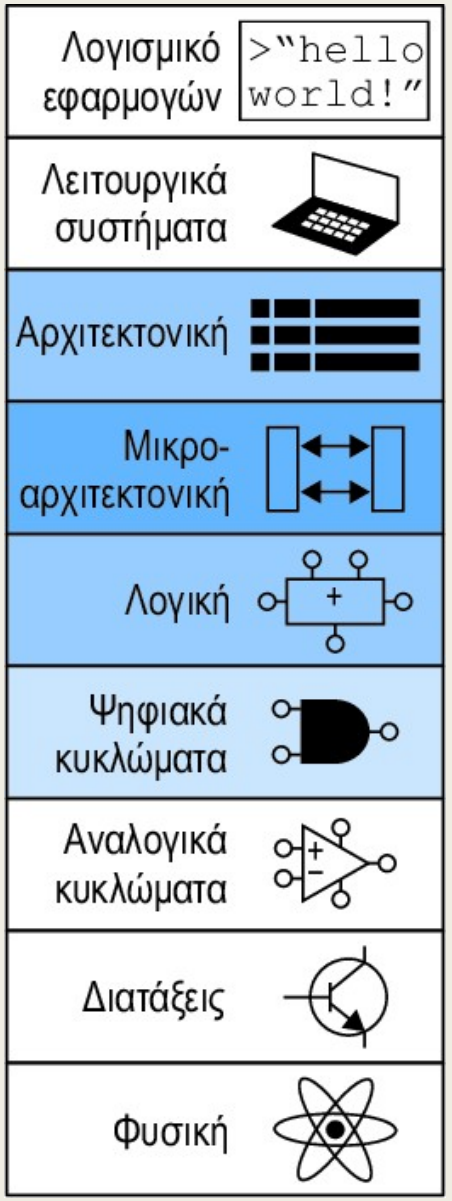
Κεφάλαιο 7

Μικροαρχιτεκτονική

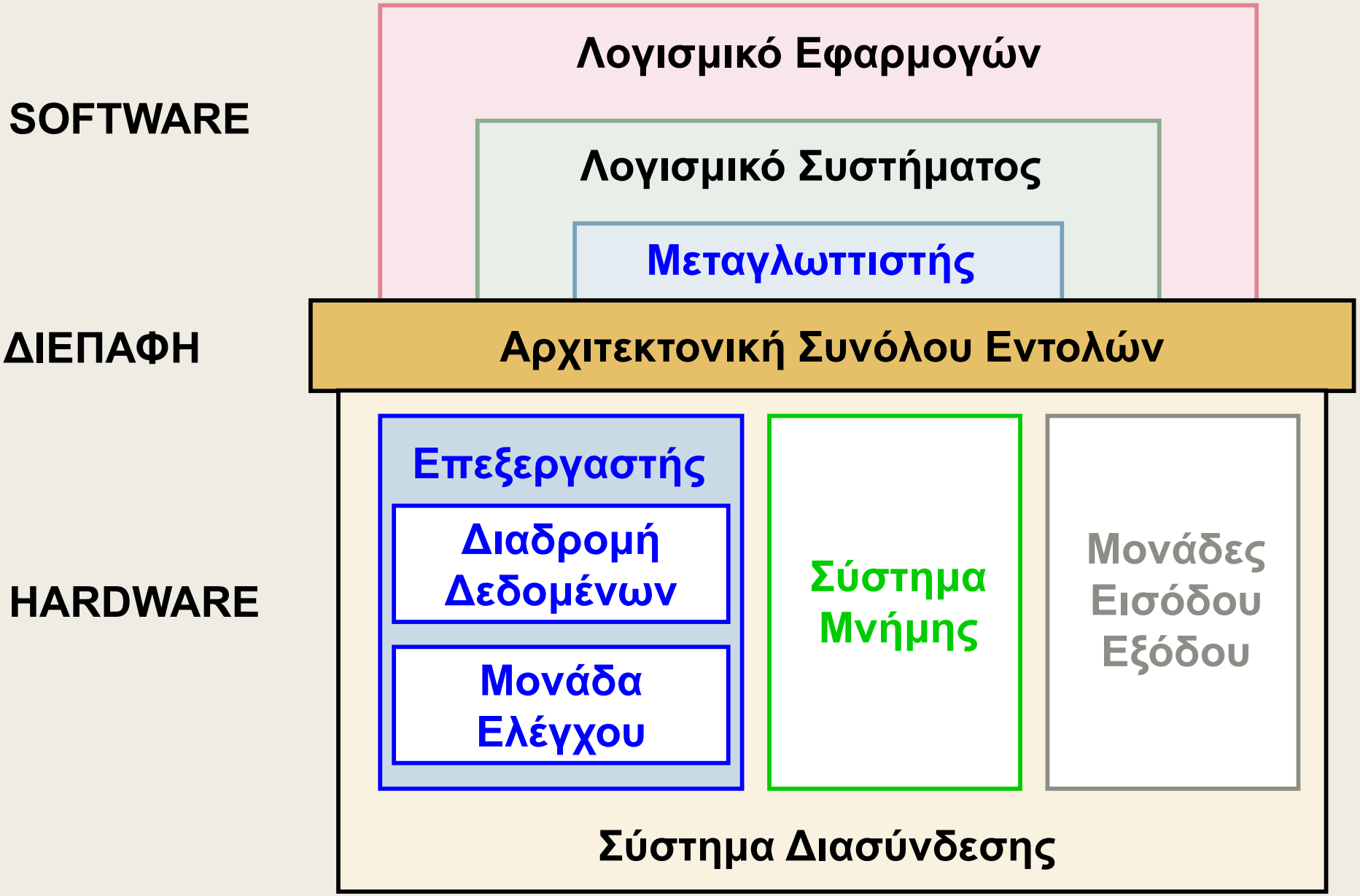
Καθηγητής Αντώνης Πασχάλης

Περιεχόμενα κεφαλαίου 7

- Εισαγωγή
- Αρχιτεκτονικοί καταχωρητές
- Βασικές μικροαρχιτεκτονικές RISC-V
- Σχεδίαση μικροαρχιτεκτονικής ενός κύκλου
  - *διαδρομή δεδομένων (datapath)*
  - *μονάδα ελέγχου (control unit)*
  - *ανάλυση επιδόσεων*



Ο υπολογιστής

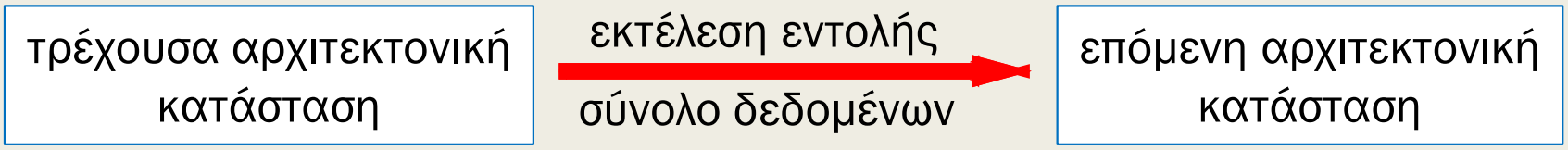


Μικροαρχιτεκτονική (microarchitecture)

- 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 κατάστασή 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 κατάστασή 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*

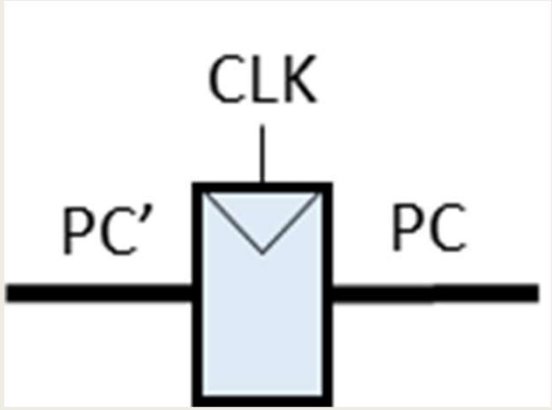


Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- 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 επεκτάσεών του (funct3 και funct7) της εντολής

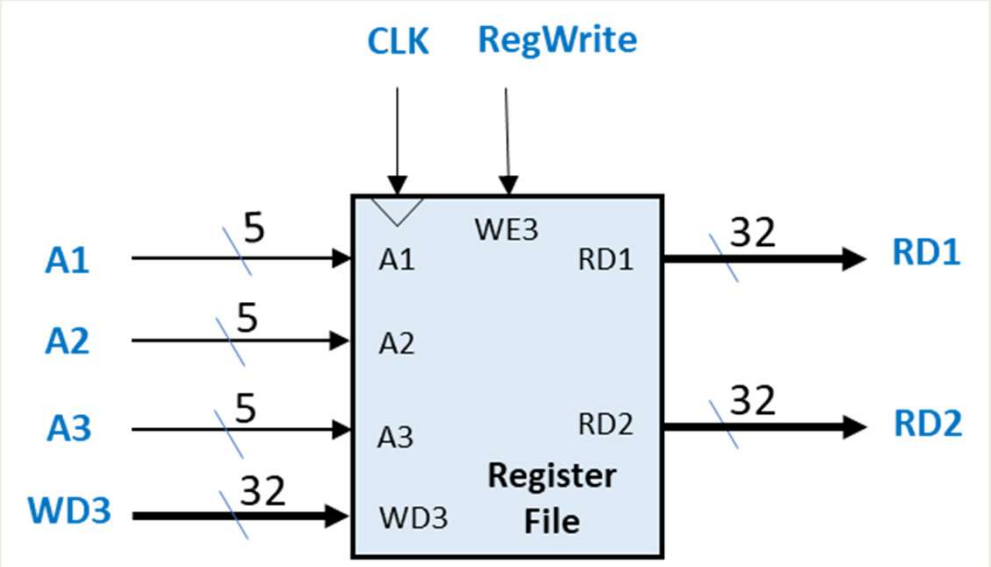
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- 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

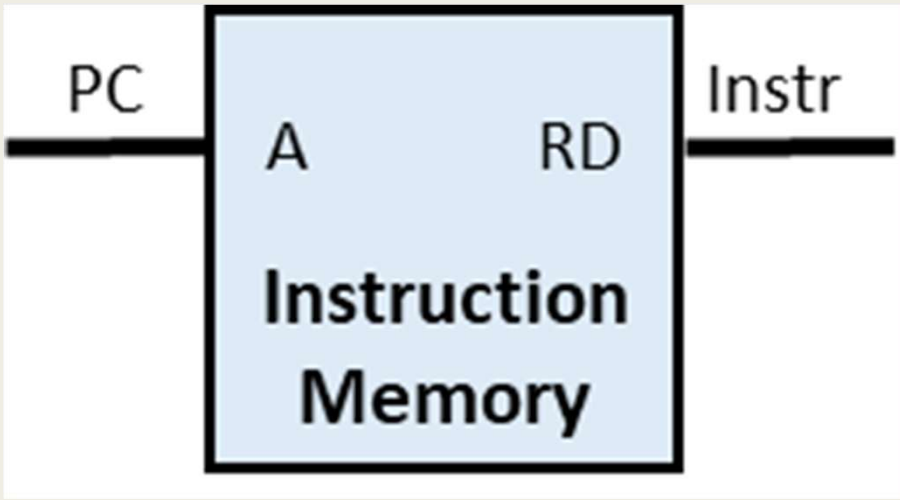
Προγραμματισμός – Κλήσεις συναρτήσεων

- 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 καταχωρητής: t0–t6
Δείκτης στοίβας: sp	Καταχωρητές ορισμάτων: a0–a7
Διεύθυνση επιστροφής: ra	
Στοίβα πάνω από τον δείκτη στοίβας	Στοίβα κάτω από τον δείκτη στοίβας

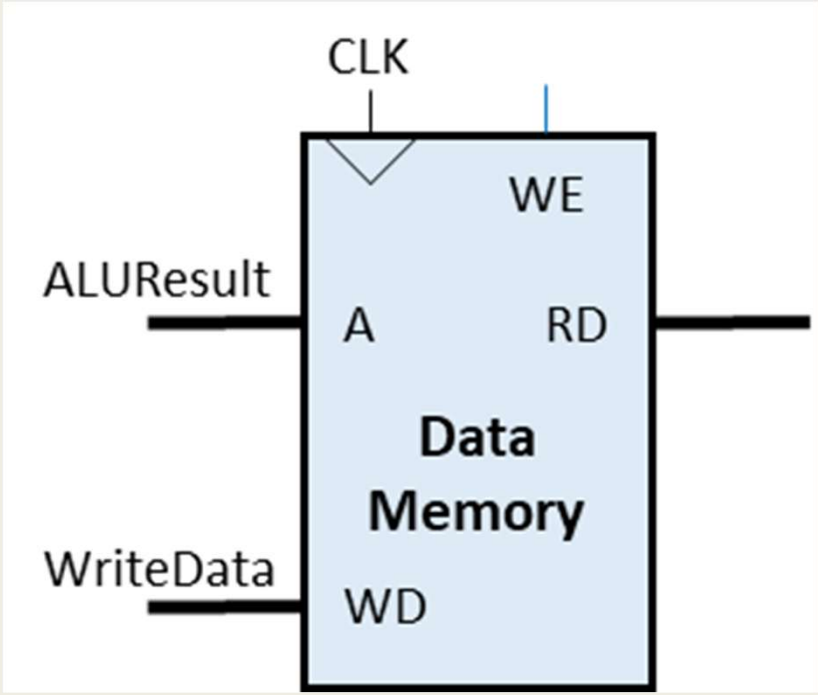
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- 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) = 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

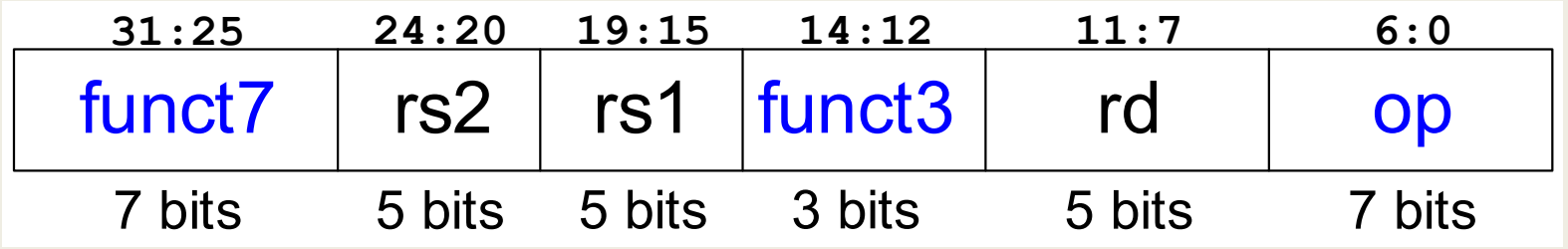
- Μικροαρχιτεκτονική πολλών κύκλων
  - *διαδρομή δεδομένων πολλών κύκλων*
  - *μονάδα ελέγχου ως μηχανή πεπερασμένων καταστάσεων (FSM)*
  - *χρήση αρχιτεκτονικών, αλλά και μη αρχιτεκτονικών καταχωρητών*
    - προσωρινοί καταχωρητές που τοποθετούνται ανάμεσα σε τμήματα συνδυαστικής λογικής, ώστε να διαιρεθεί 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 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)*

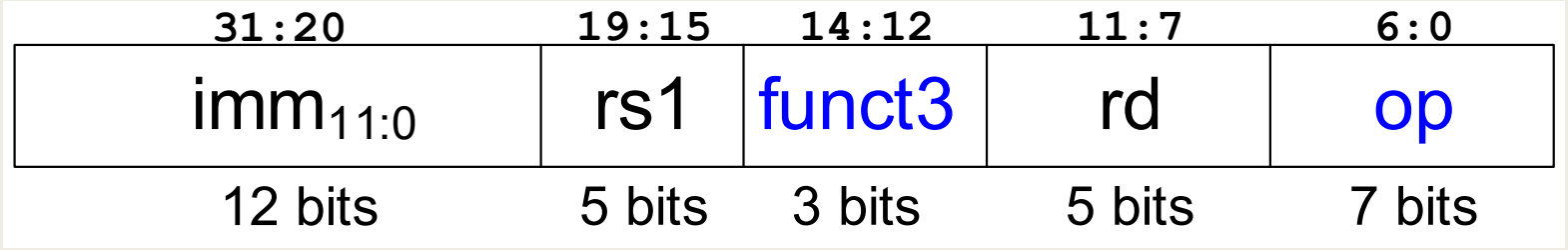


- `ADD  Rd, Rs1, Rs2` `Rd = Rs1 + Rs2`
- `SUB  Rd, Rs1, Rs2` `Rd = Rs1 - Rs2`
- `AND  Rd, Rs1, Rs2` `Rd = Rs1 and Rs2`
- `OR  Rd, Rs1, Rs2` `Rd = Rs1 or  Rs2`
- `XOR  Rd, Rs1, Rs2` `Rd = Rs1 xor Rs2`
- `SLT  Rd, Rs1, Rs2` `Rd = 1 if rs1<rs2, else 0`
- `SLTU Rd, Rs1, Rs2` `Rd = 1 if rs1<rs2, else 0`
- `SLL  Rd, Rs1, Rs2` `Rd = Rs1 <<  Rs2(4:0)`
- `SRL  Rd, Rs1, Rs2` `Rd = Rs1 >>  Rs2(4:0)`
- `SRA  Rd, Rs1, Rs2` `Rd = Rs1 >>> 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)*

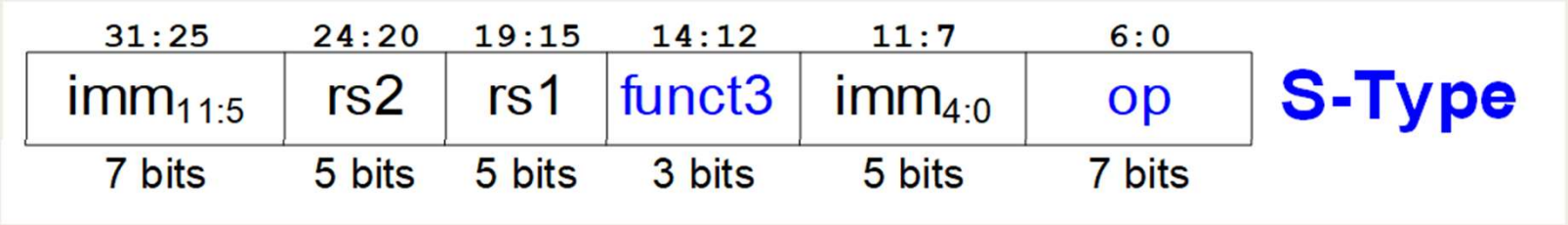


- **ADDI Rd, Rs1, imm** — **Rd = Rs1 + S_Ext(imm)**
- **ANDI Rd, Rs1, imm** — **Rd = Rs1 and S_Ext(imm)**
- **ORI Rd, Rs1, imm** — **Rd = Rs1 or S_Ext(imm)**
- **XORI Rd, Rs1, imm** — **Rd = Rs1 xor S_Ext(imm)**
- **LW Rd, imm(Rs1)** — **Rd = mem[Rs1+S_Ext(imm)]**
- **SLLI Rd, Rs1, uimm** — **Rd = Rs1 << imm(4:0)**
- **SRLI Rd, Rs1, uimm** — **Rd = Rs1 >> imm(4:0)**
- **SRAI Rd, Rs1, uimm** — **Rd = Rs1 >>> 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

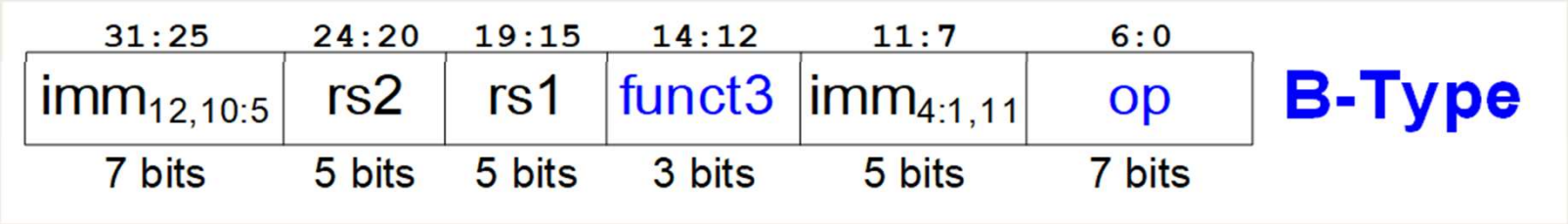
- Στη συνέχεια, επιλέγουμε ποιες βασικές εντολές θα υλοποιήσουμε:
  - *Εντολές τύπου S (store)*



  - **`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 + S_Ext (imm(12:1)x2)

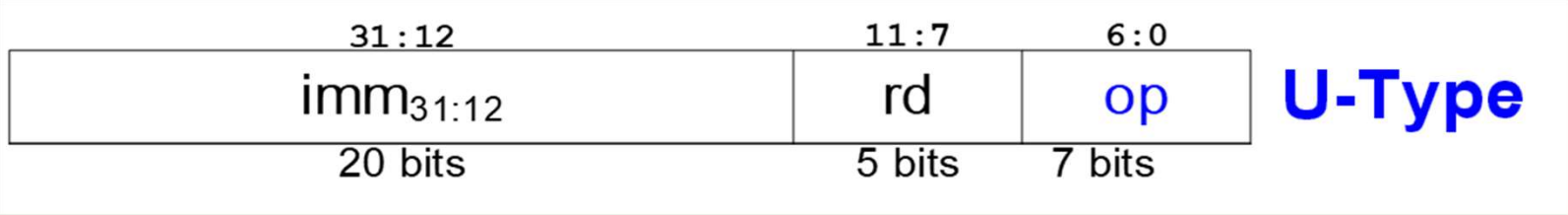
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- 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 ≠ 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 ≥ 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 ≤ 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

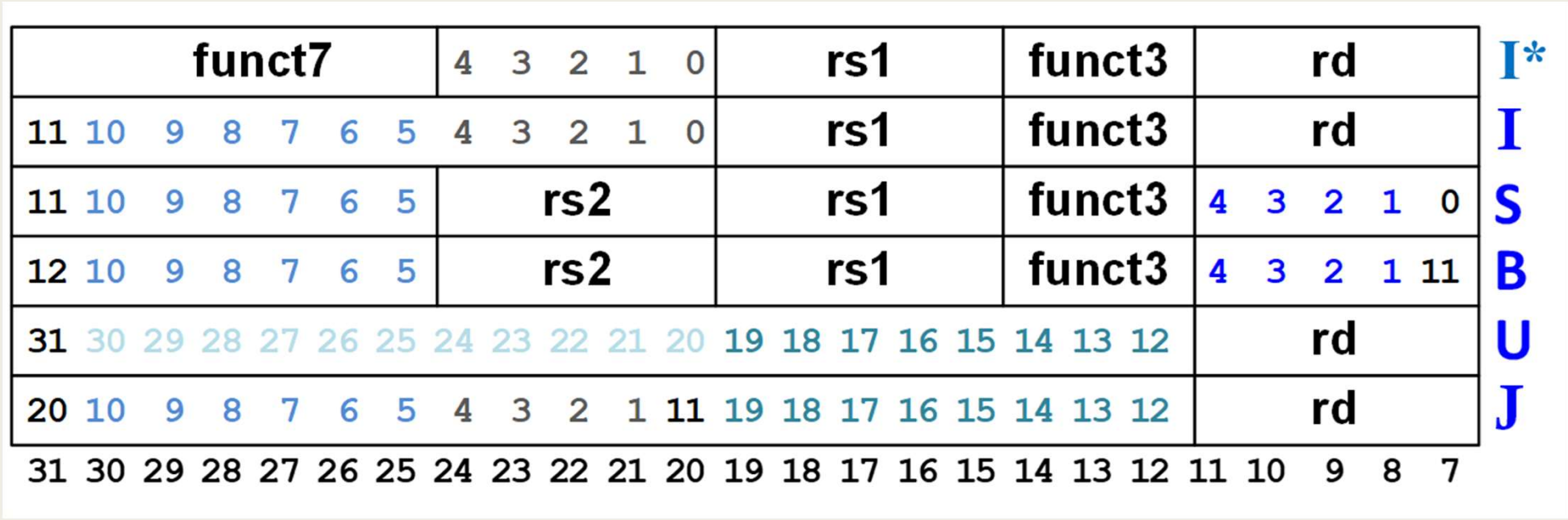
Σχεδιαστική διαδικασία μικροαρχιτεκτονικής RISC-V

- 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 έμμεσα)	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



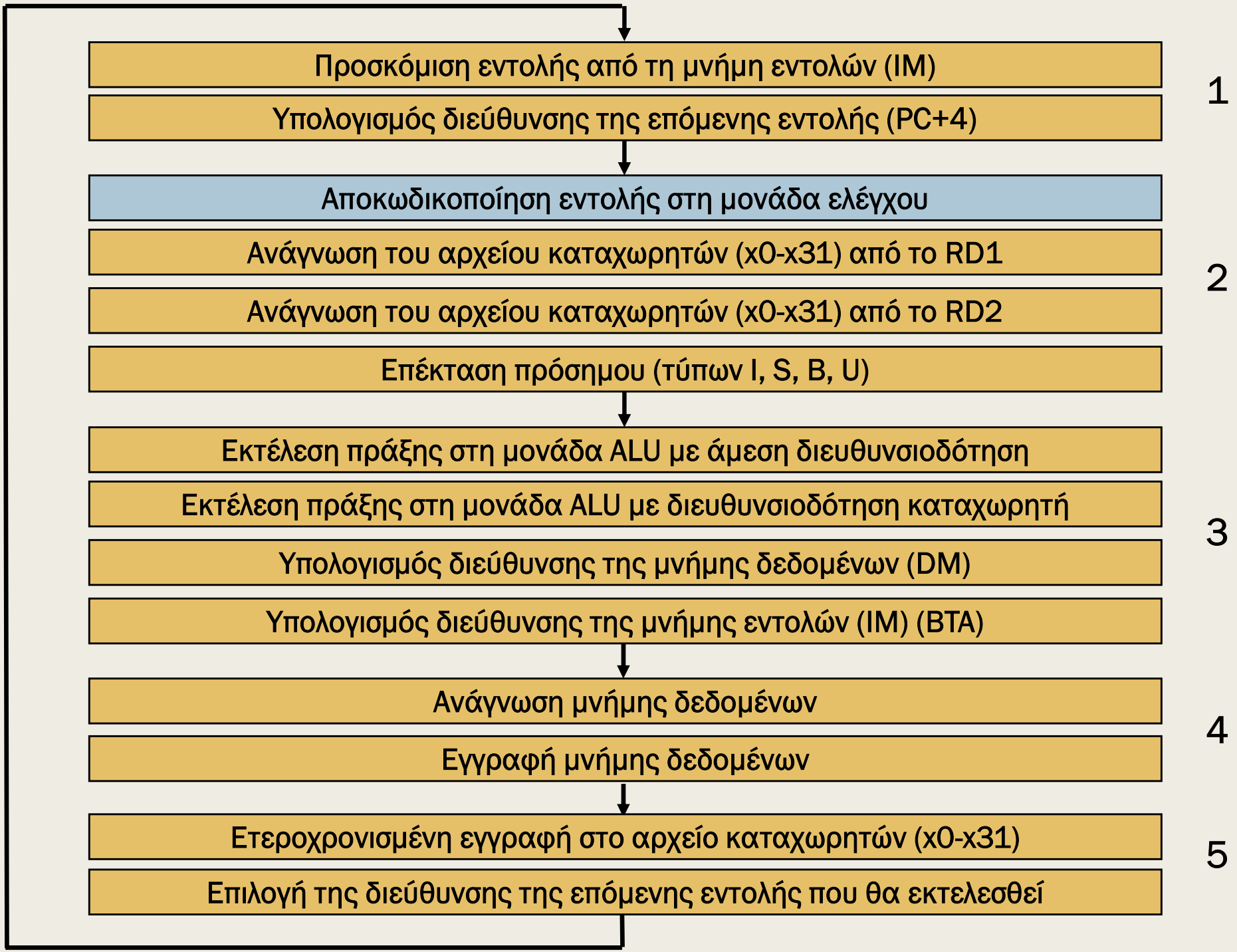
- Τα bit του άμεσου τελεστέου αποθηκεύονται στις περισσότερες των περιπτώσεων στη σειρά σε κατάλληλα πεδία της εντολής
  - *Απλοποιεί το υλικό για την κατασκευή του μικροεπεξεργαστή*
- Οι άμεσοι τελεστέοι υπόκεινται σε **επέκταση προσήμου στα 32 bit** πριν τη χρήση τους
  - *το bit προσήμου του άμεσου 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 επόμενης διεύθυνσης (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)

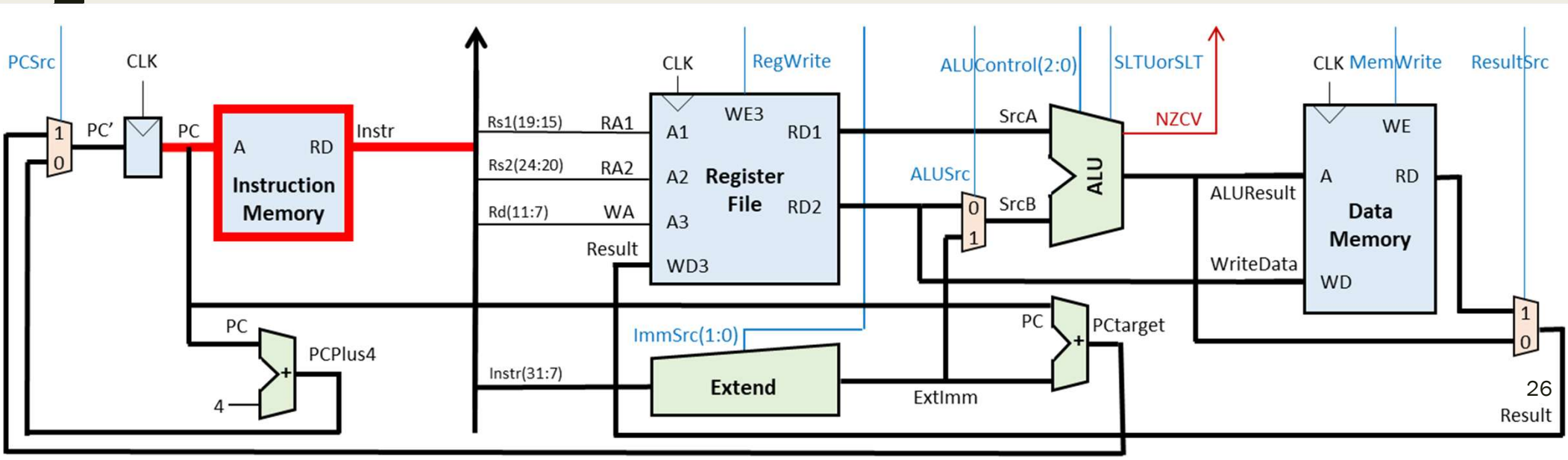


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων

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 x 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

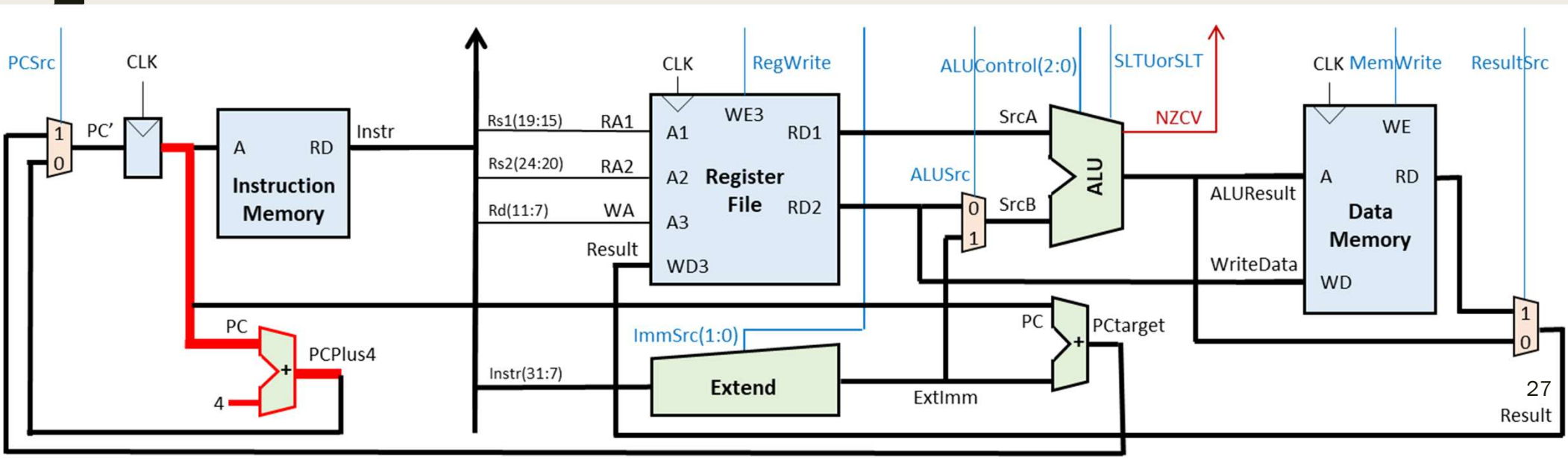


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων

Υπολογισμός διεύθυνσης της επόμενης εντολής (PC+4)

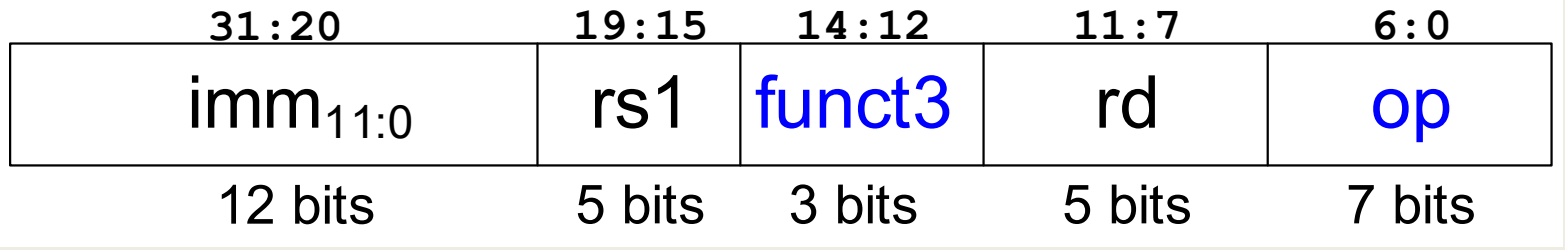
- Υπολογισμός της διεύθυνσης της αμέσως επόμενης εντολής PC+4 (παράλληλα με την εκτέλεση της τρέχουσας εντολής)
  - *σύνδεση της εξόδου PC του μετρητή προγράμματος με την είσοδο του αθροιστή κατά 4 (incrementer)*
  - *εμφάνιση του αποτελέσματος της αύξησης κατά 4 στην έξοδο PCPlus4 του αθροιστή κατά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

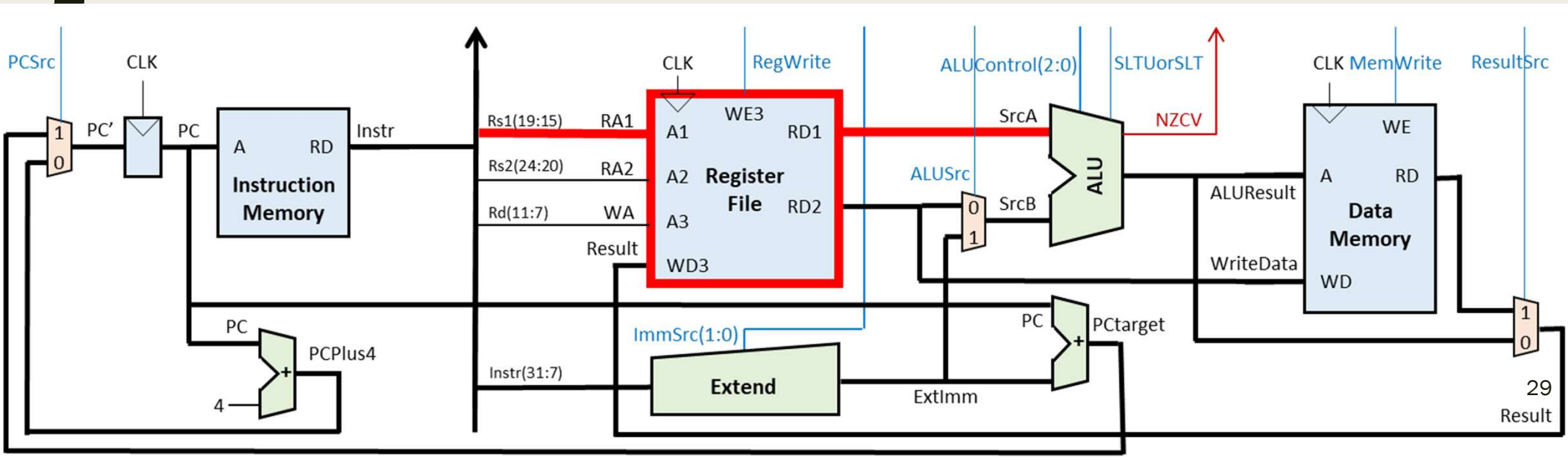
- Αρχικά, θα μελετήσουμε τη διαδρομή δεδομένων που υλοποιεί την εντολή LW
- Κώδικας συμβολικής γλώσσας:
  - **`LW Rd, imm(Rs1)  Rd = mem[Rs1+S_Ext(imm)]`**
- Στη μορφή της εντολής ορίζονται:
  - *οι διευθύνσεις του καταχωρητή προέλευσης Rs1 και του καταχωρητή προορισμού Rd*
  - *ένας προσημασμένος άμεσος 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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

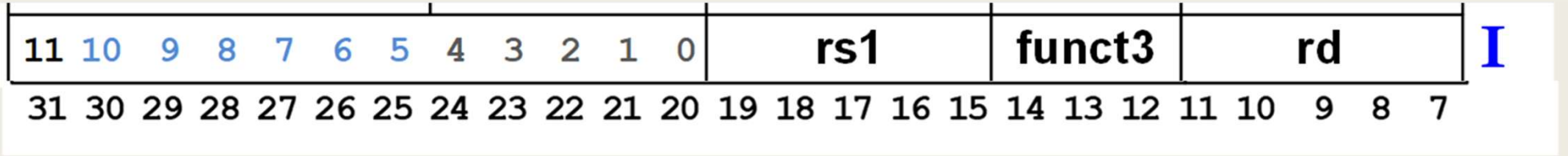
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

- 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)

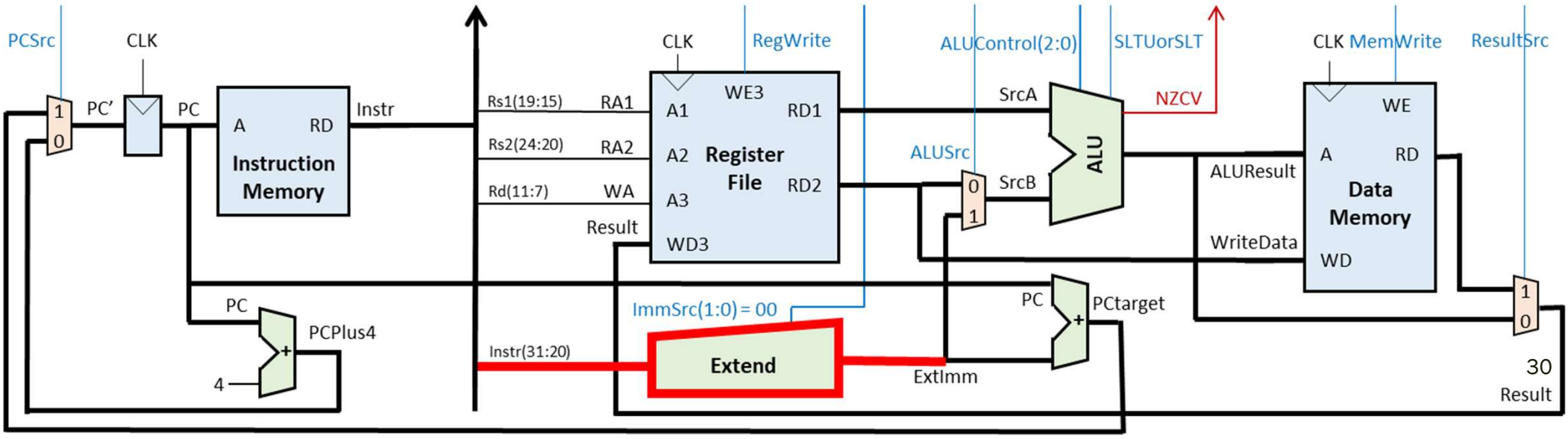


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

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



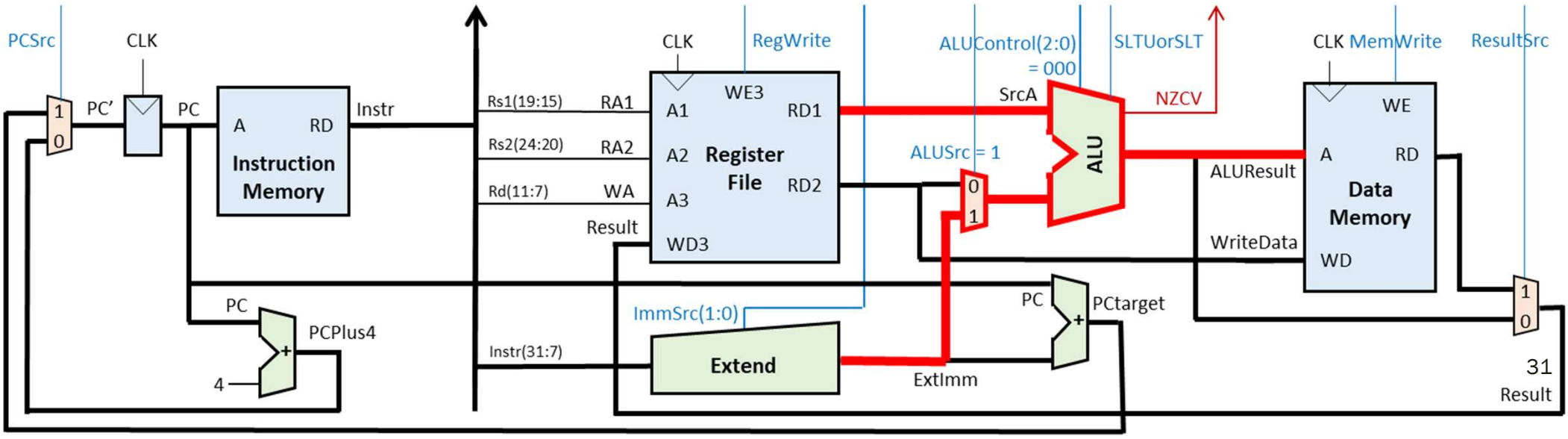
- Η εντολή LW απαιτεί και μια **σχετική απόσταση (offset)** για τον υπολογισμό της διεύθυνσης της μνήμης δεδομένων που προκύπτει με επέκταση προσήμου από τα 12 bit στα 32 bit στην μονάδα **Extend**
  - **`Extend(31:0) = Sign_Ext(instr(31:20))`**
  - *Για επέκταση προσήμου τύπου I τα σήματα ελέγχου είναι:*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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

Υπολογισμός διεύθυνσης της μνήμης δεδομένων (DM)

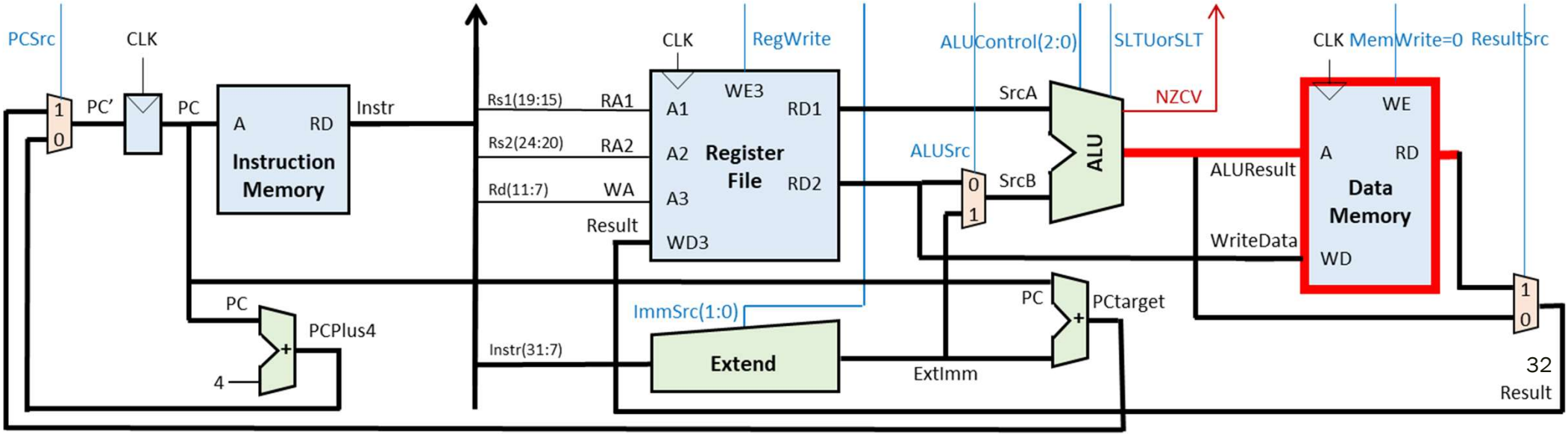
- Η σχετική απόσταση προστίθεται στη διεύθυνση βάσης στη μονάδα ALU, ώστε να προκύψει η διεύθυνση της μνήμης δεδομένων ως ALUResult
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA των 32 bit της μονάδας ALU*
  - *σύνδεση της εξόδου ExtImm της μονάδας Extend με την είσοδο SrcB των 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

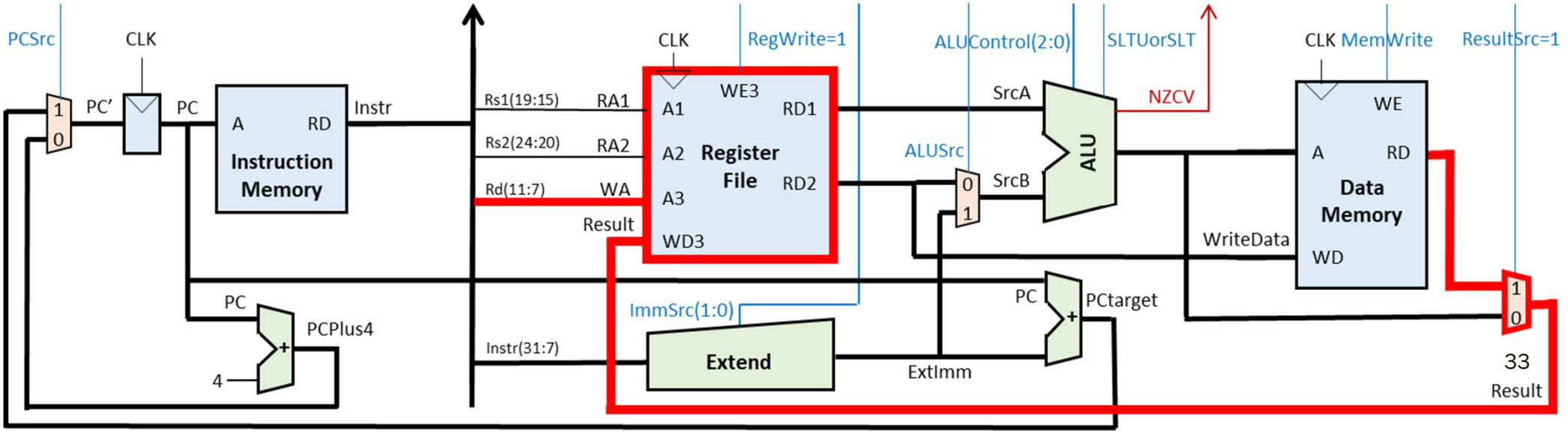
- Η διεύθυνση που εμφανίζεται στην έξοδο ALUResult της μονάδας ALU διευθυνσιοδοτεί την είσοδο διευθύνσεων A της μνήμης δεδομένων (DM)
- Η λέξη δεδομένων διαβάζεται σύγχρονα (Block RAM) ή ασύγχρονα (Distributed RAM) στη θύρα ανάγνωσης RD της DM των 2^M x 32 bit
  - *η διεύθυνση της λέξης δεδομένων είναι 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)

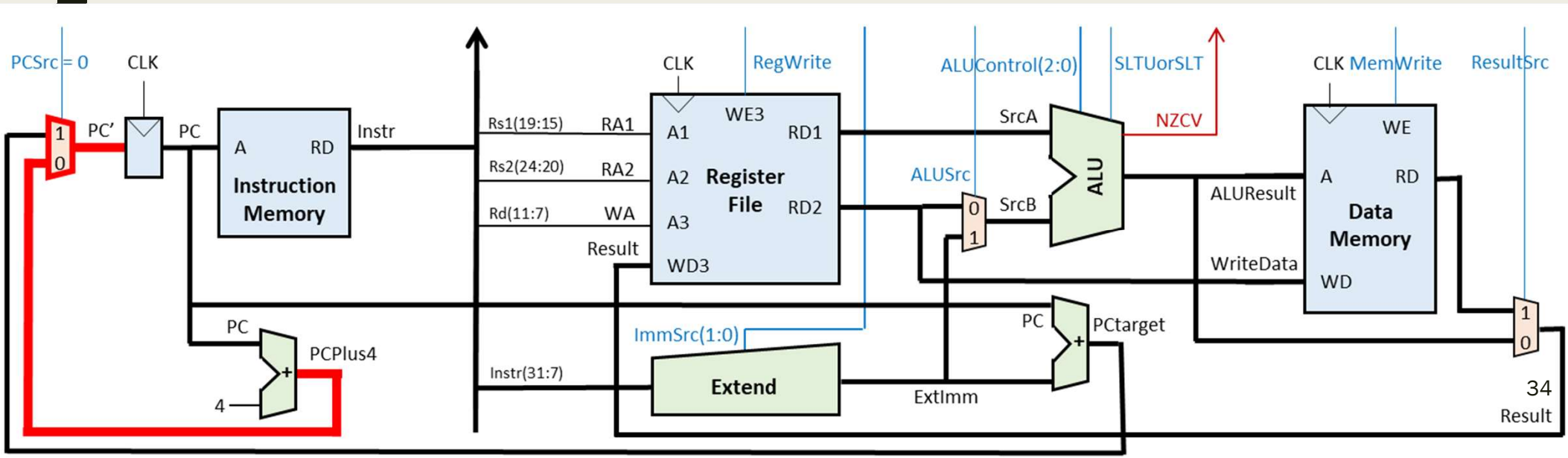
- Η **λέξη δεδομένων** που διαβάζεται από τη μνήμη δεδομένων εγγράφεται ετερο-χρονισμένα στον **καταχωρητή προορισμού Rd** του αρχείου καταχωρητών
  - *σύνδεση του πεδίου Rd της εντολής (Instr11:7) στην είσοδο διευθύνσεων εγγραφής A3*
  - *σύνδεση της θύρας ανάγνωσης RD της μνήμης δεδομένων με τη θύρα 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

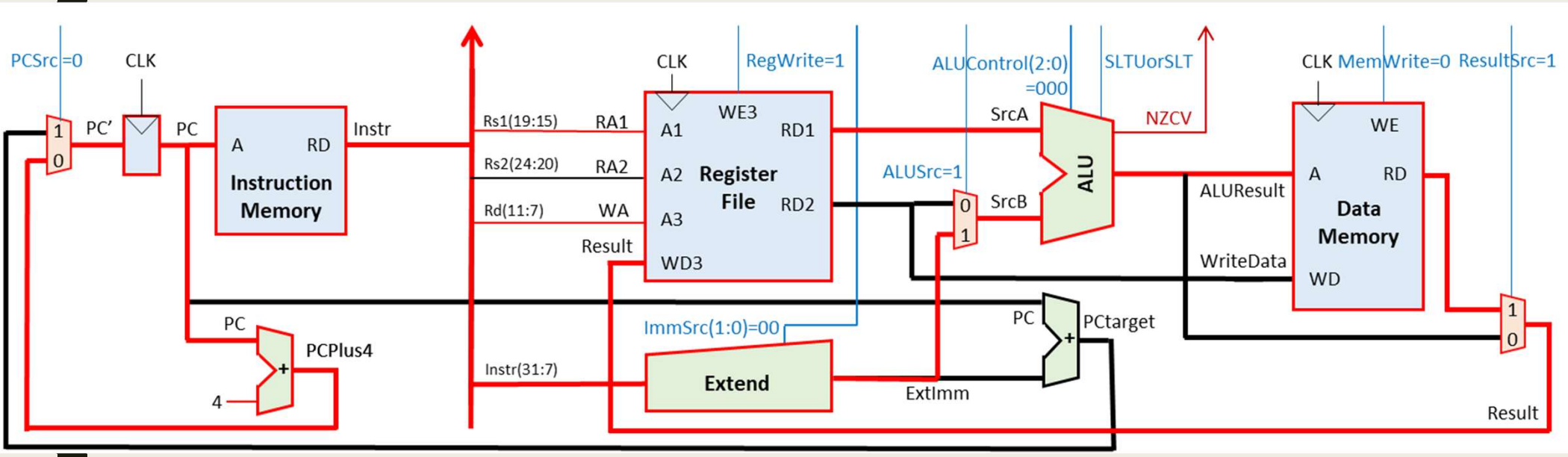
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- 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

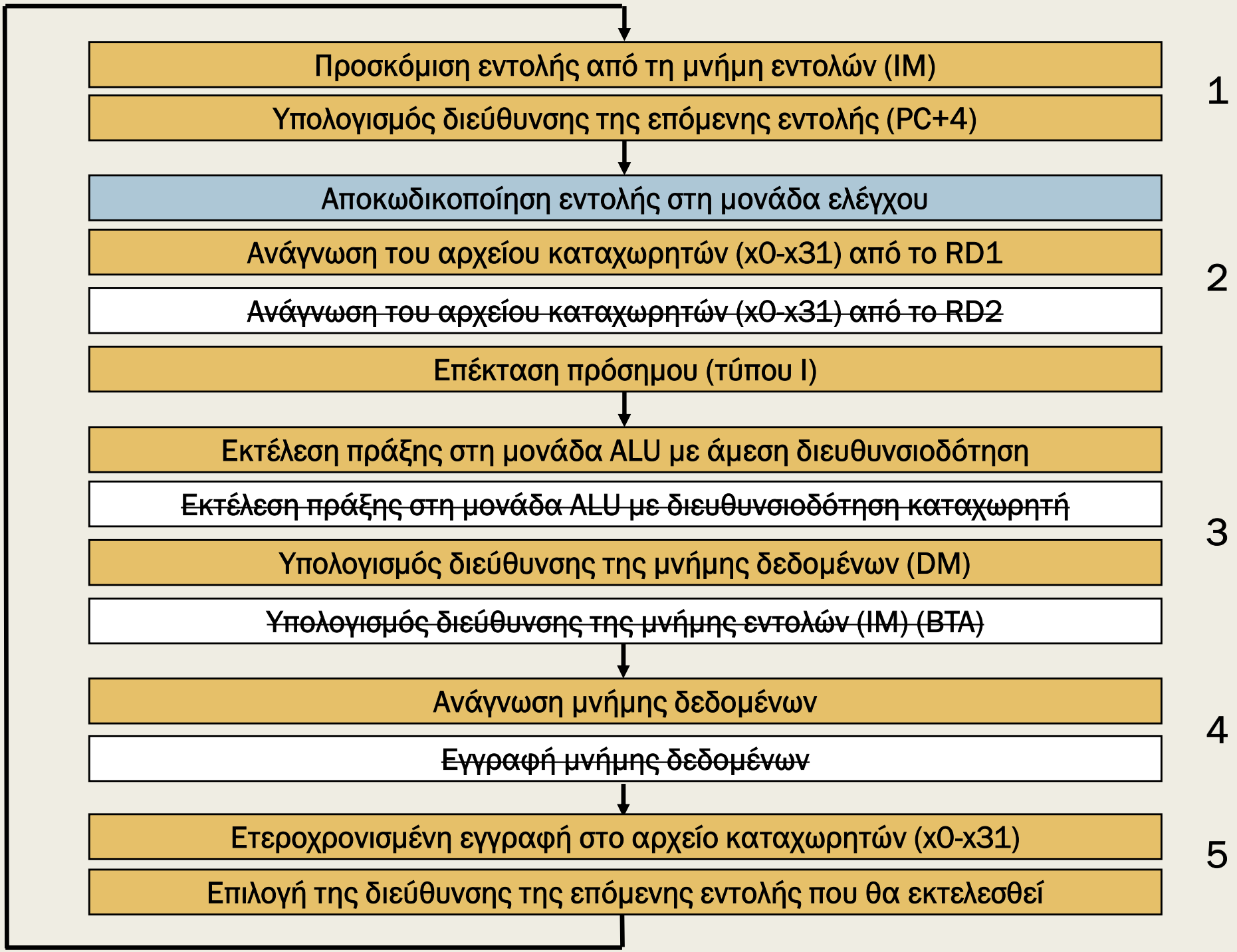


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή LW

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

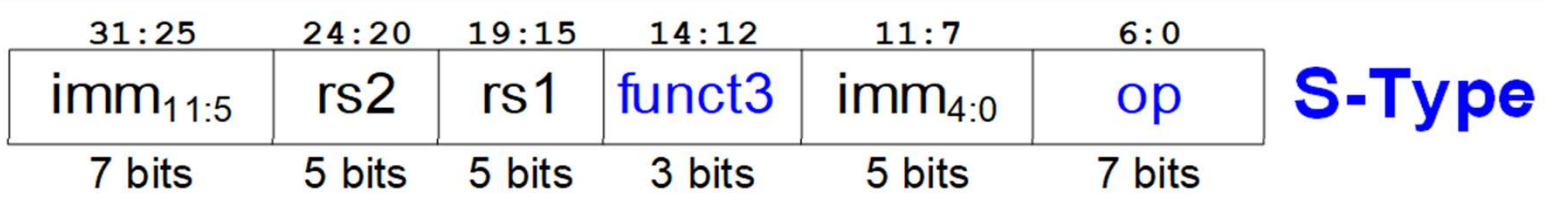


Λειτουργίες εντολής LW (10)



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



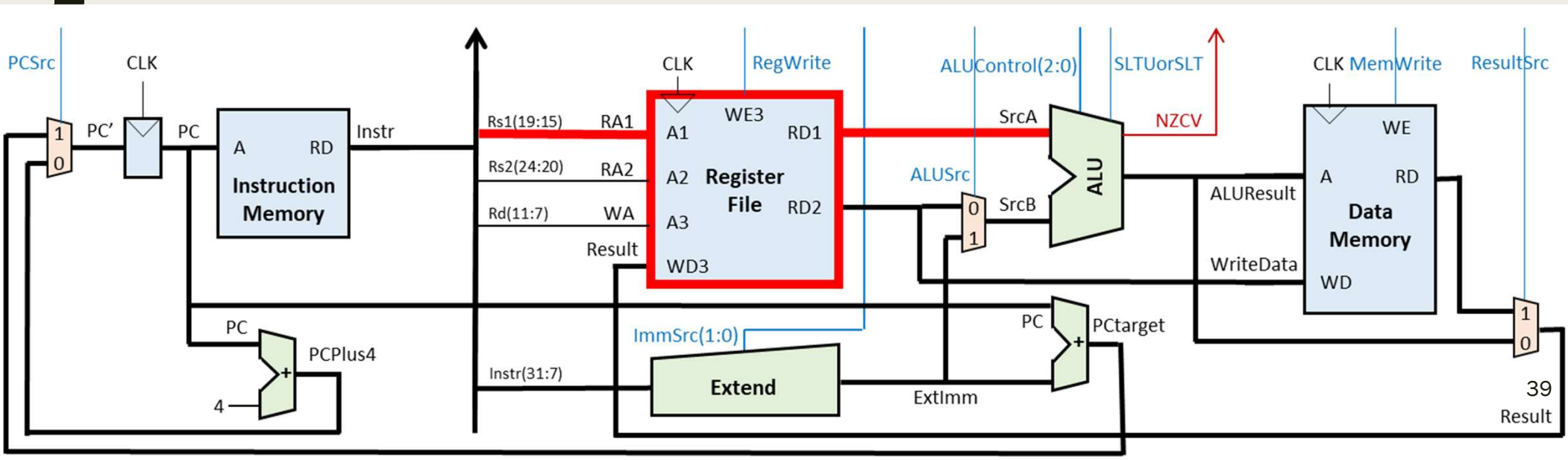
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

- 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

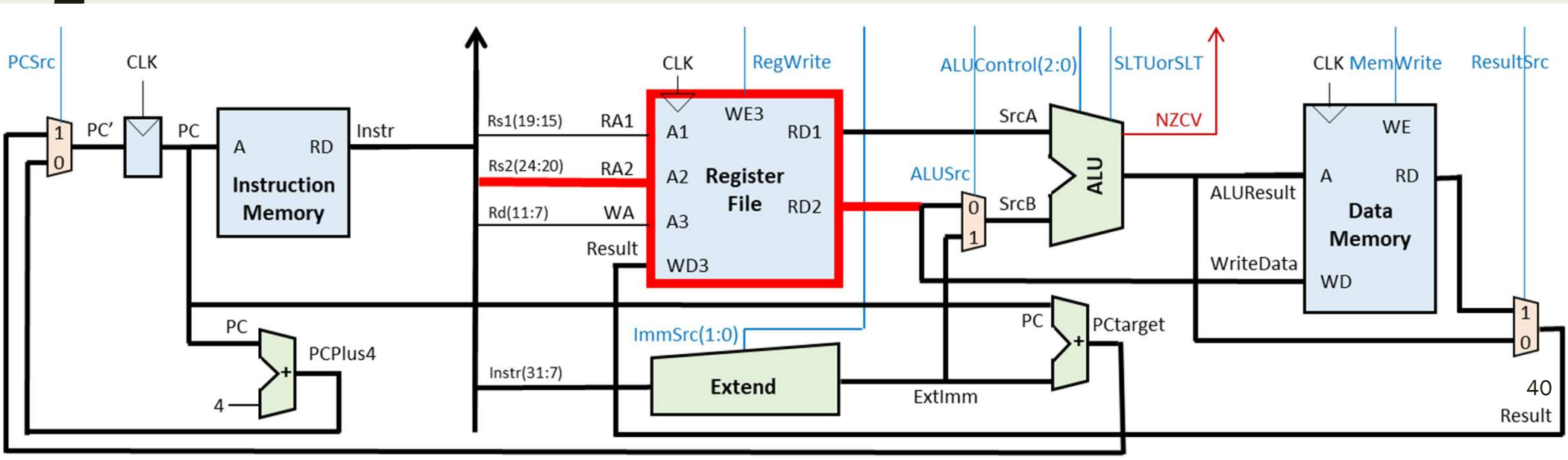
- Η διεύθυνση βάσης που είναι αποθηκευμένη στον καταχωρητή προέλευσης Rs1 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
  - *σύνδεση του πεδίου Rs1 της εντολής (Instr19:15) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης 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

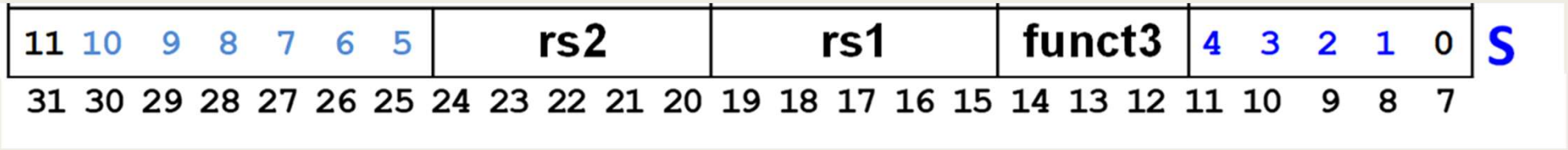
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

- 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)

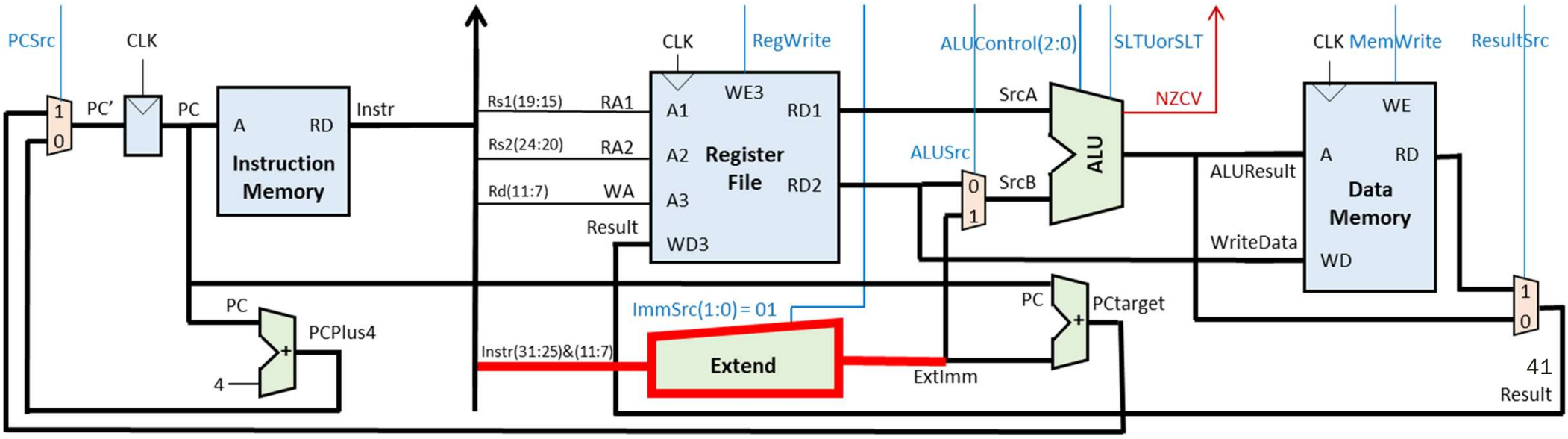


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

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



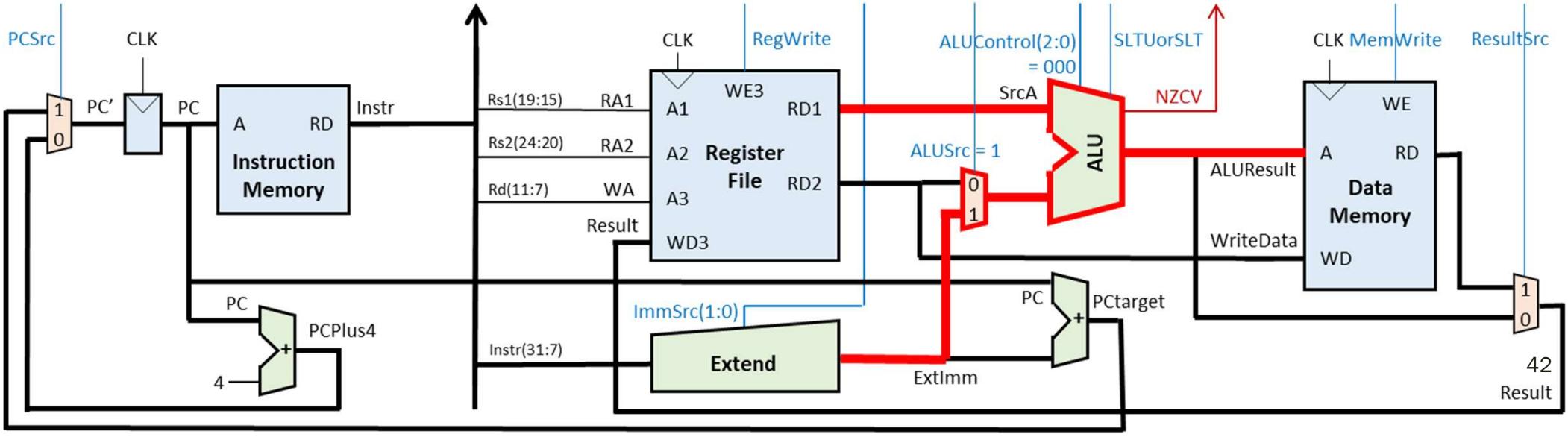
- Η εντολή SW απαιτεί και μια σχετική απόσταση (offset) για τον υπολογισμό της διεύθυνσης της μνήμης δεδομένων που προκύπτει με επέκταση προσήμου από τα 12 bit στα 32 bit στην μονάδα Extend
  - **`Extend(31:0) = Sign_Ext(instr(31:25)&instr(11:7))`**
  - *Για επέκταση προσήμου τύπου S τα σήματα ελέγχου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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

Υπολογισμός διεύθυνσης της μνήμης δεδομένων (DM)

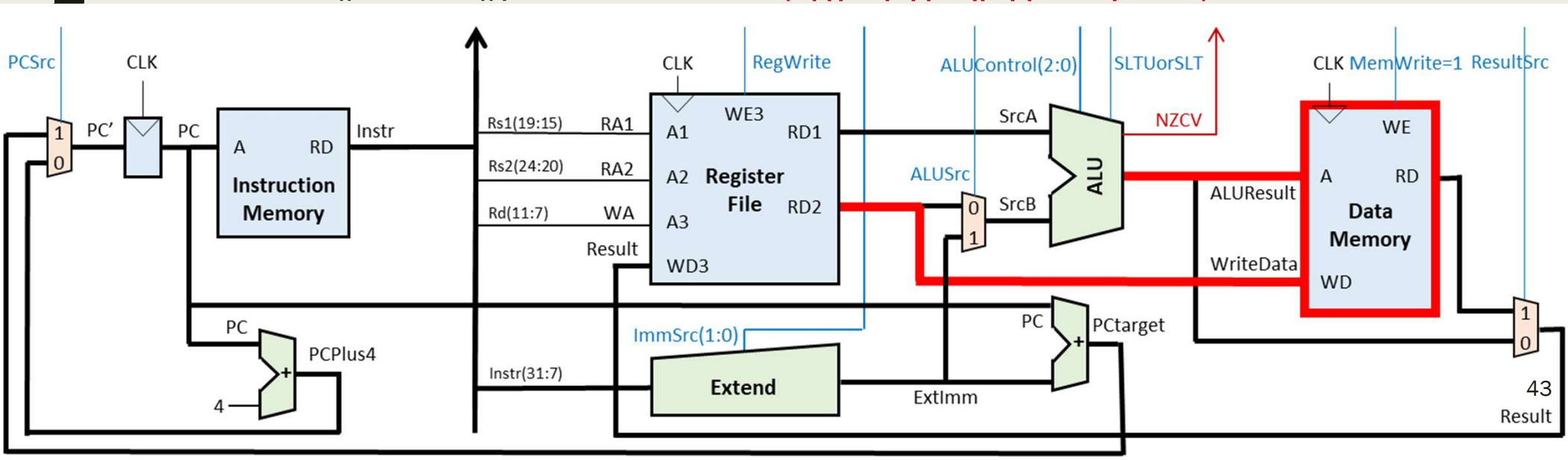
- Η σχετική απόσταση προστίθεται στη διεύθυνση βάσης στη μονάδα ALU, ώστε να προκύψει η διεύθυνση της μνήμης δεδομένων ως ALUResult
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA των 32 bit της μονάδας ALU*
  - *σύνδεση της εξόδου ExtImm της μονάδας Extend με την είσοδο SrcB των 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

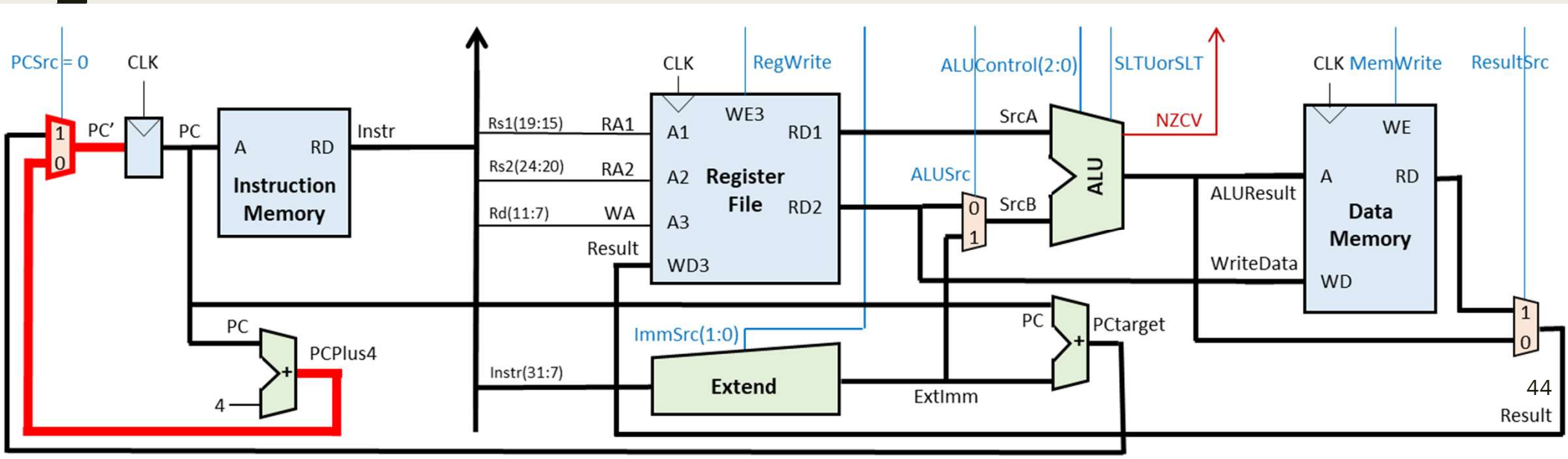
- Η διεύθυνση που εμφανίζεται στην έξοδο ALUResult της μονάδας ALU διευθυνσιοδοτεί την είσοδο διευθύνσεων A της μνήμης δεδομένων (DM)
- Η λέξη δεδομένων (WriteData), που εμφανίζεται στη θύρα ανάγνωσης RD2, γράφεται σύγχρονα στη μνήμη δεδομένων DM των 2^M x 32 bit μέσω της θύρας εγγραφής δεδομένων WD 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

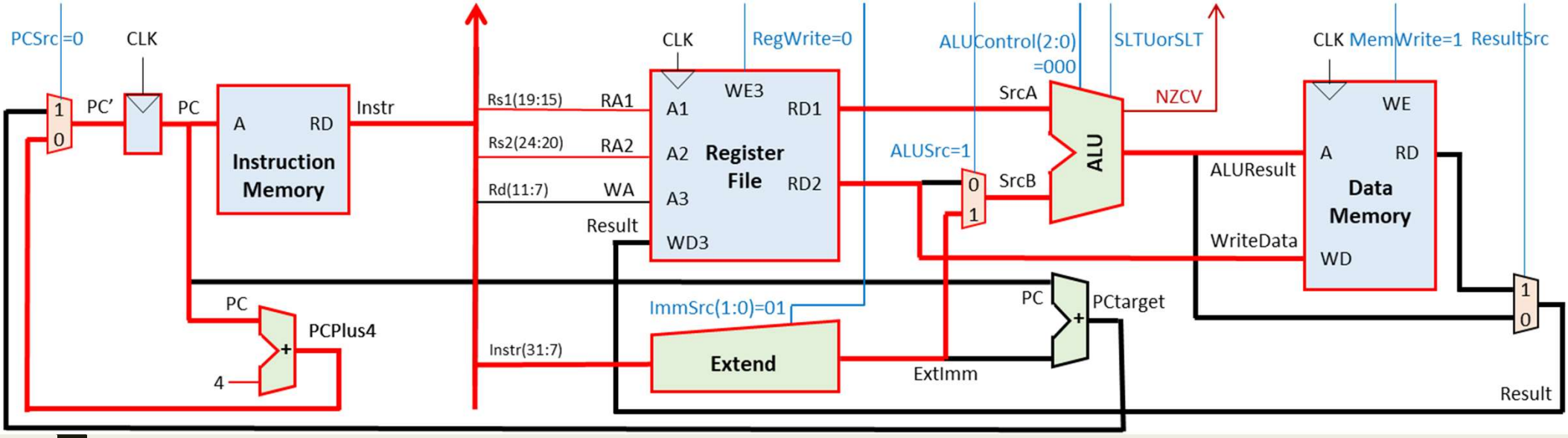
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- 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

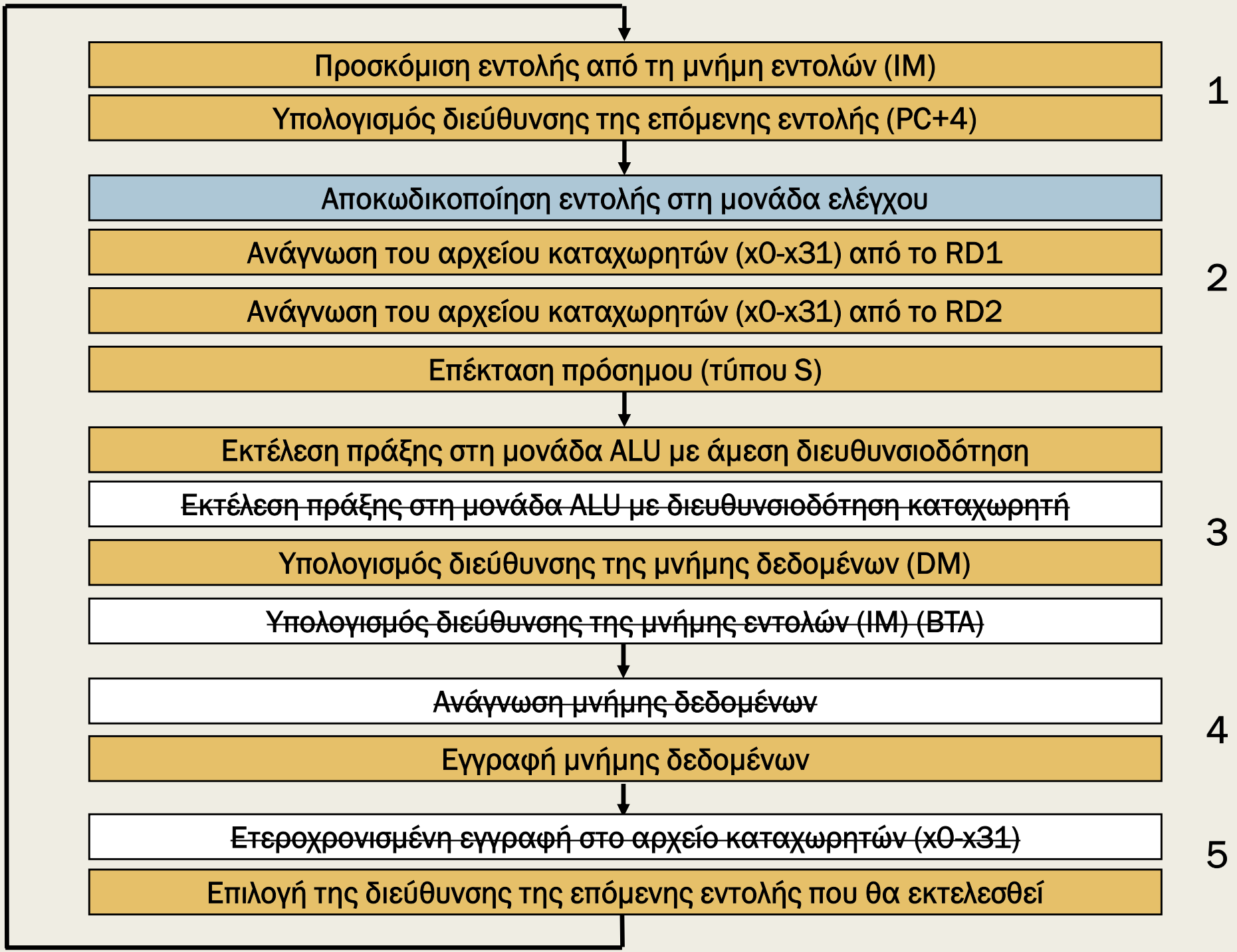


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή SW

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

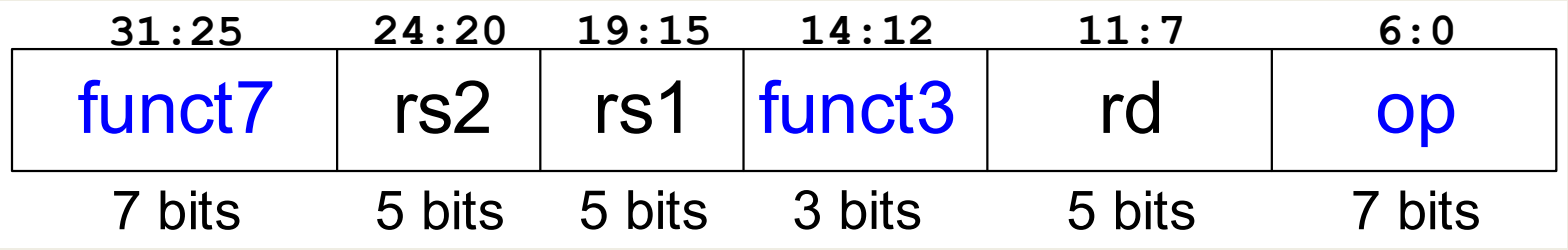


Λειτουργίες εντολής SW (10)



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*



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

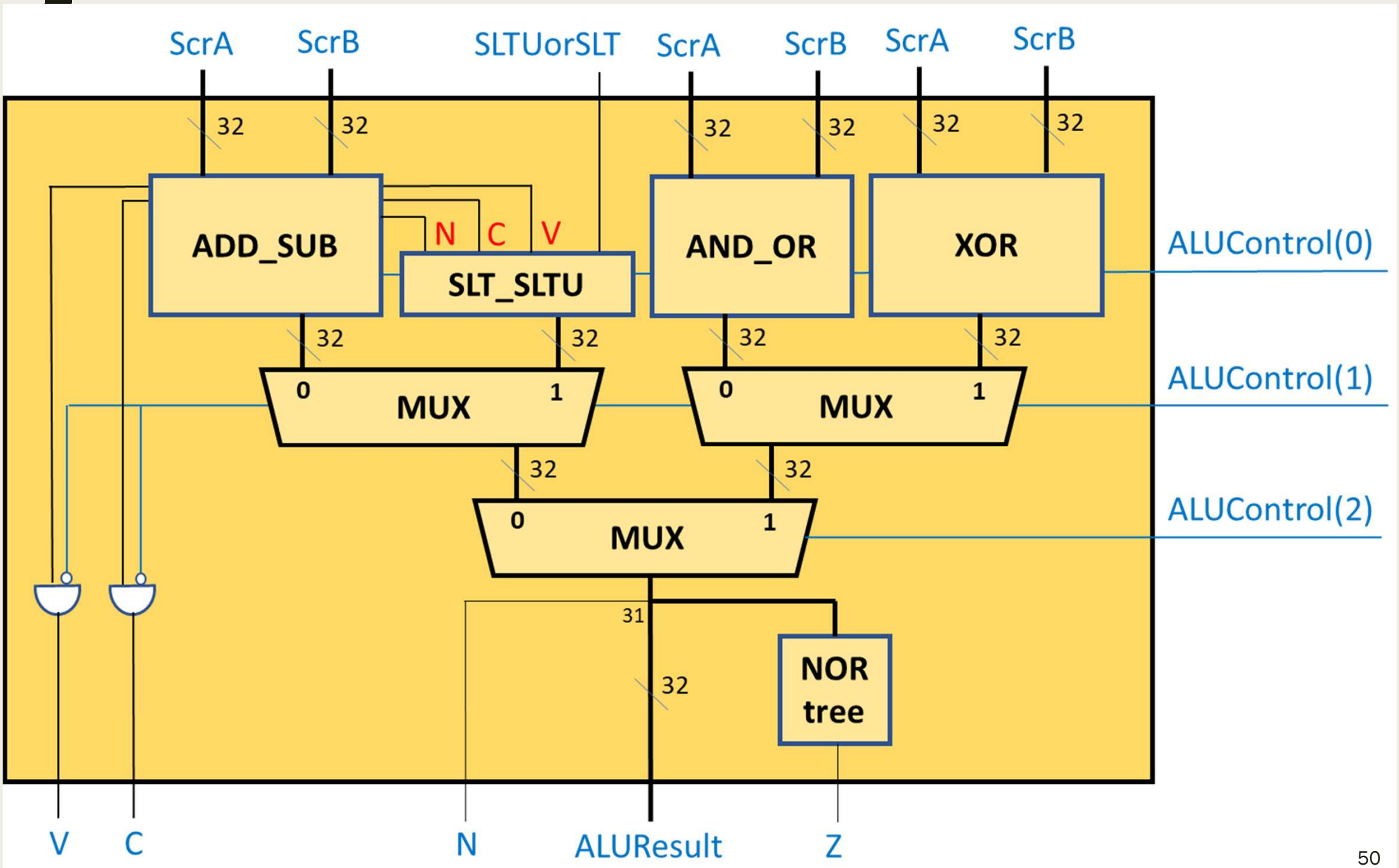
- Απαιτούμενες λειτουργίες των εντολών ALU που ήδη υποστηρίζονται:
  - *ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs1 του αρχείου καταχωρητών διαβάζεται στη θύρα ανάγνωσης RD1*
  - *ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs2 του αρχείου καταχωρητών διαβάζεται 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 σημαίες N, Z, C, V (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

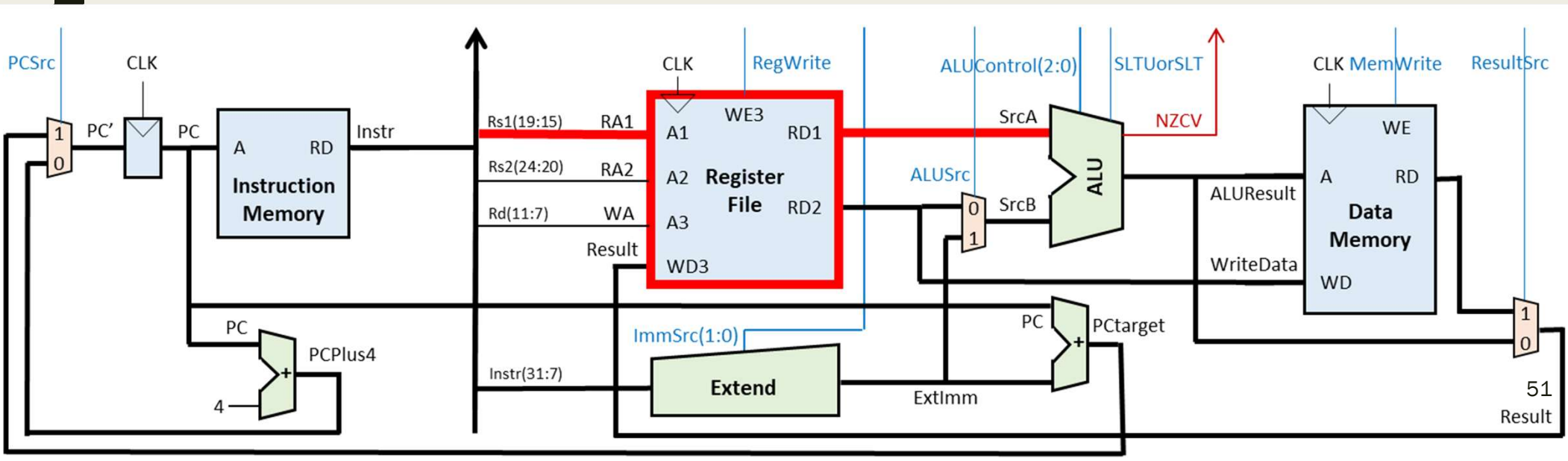
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Η μονάδα ALU



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

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

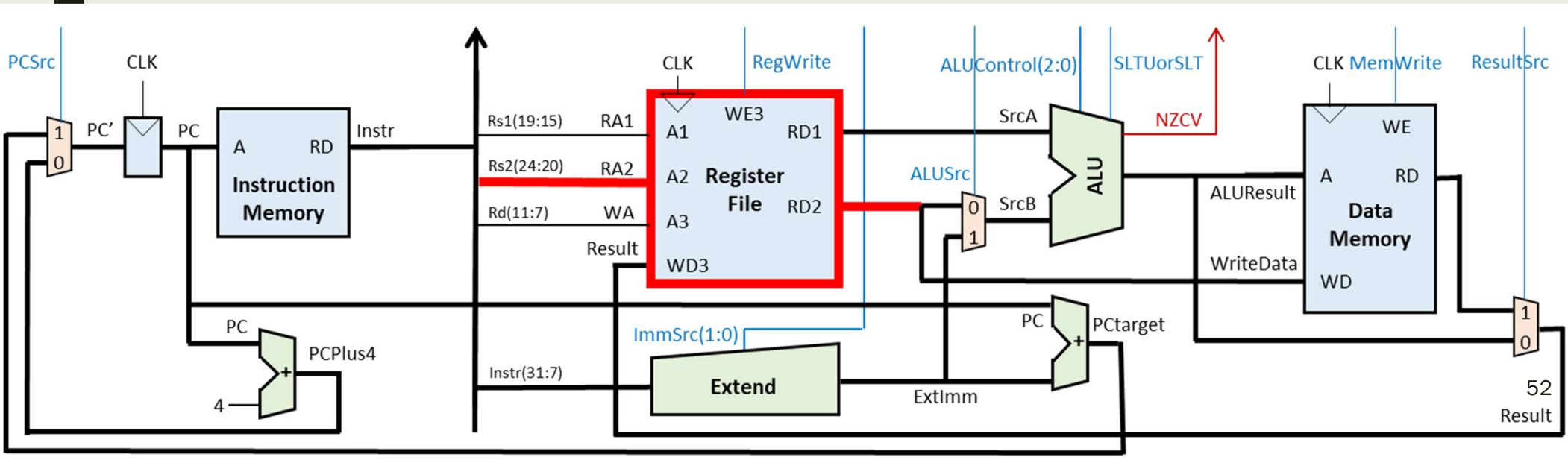
- ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs1 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
  - *σύνδεση του πεδίου Rs1 της εντολής (Instr19:15) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης RD1*
  - *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

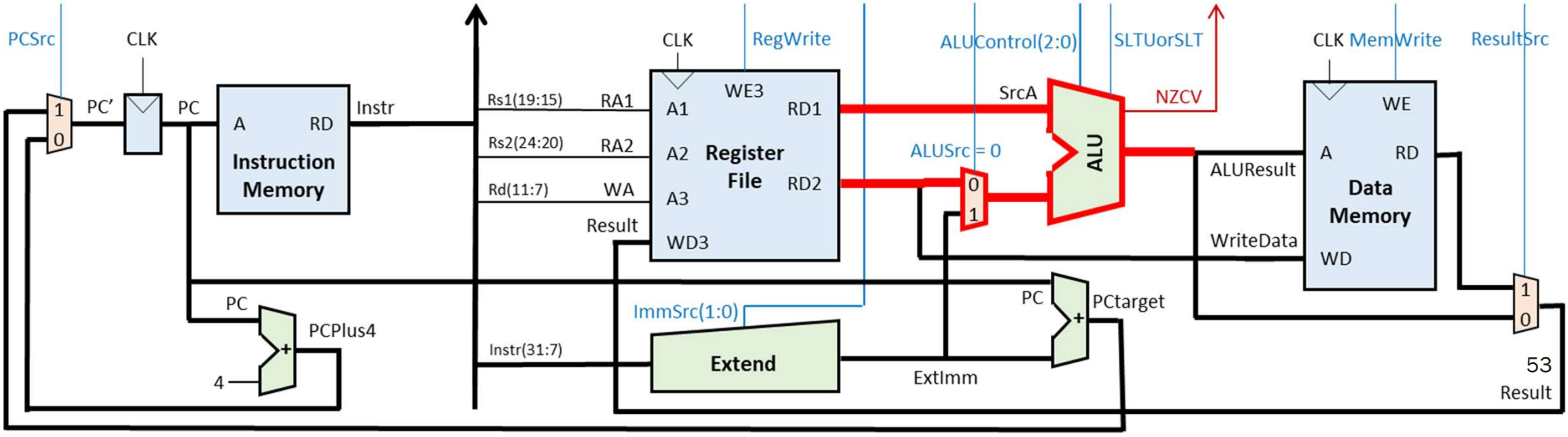
- ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs2 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD2
  - *σύνδεση του πεδίου Rs2 της εντολής (Instr24:20) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης RD2*
  - *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

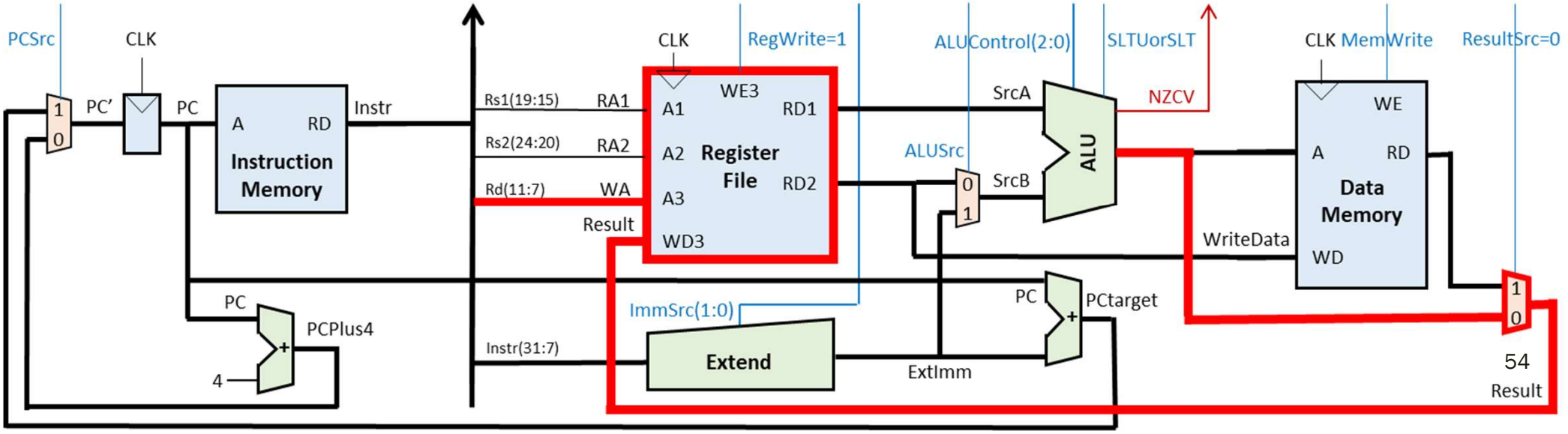
- Η μονάδα ALU εκτελεί τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR μετά από:
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA των 32 bit της μονάδας ALU*
  - *σύνδεση της θύρας RD2 του αρχείου καταχωρητών με την είσοδο SrcB 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)

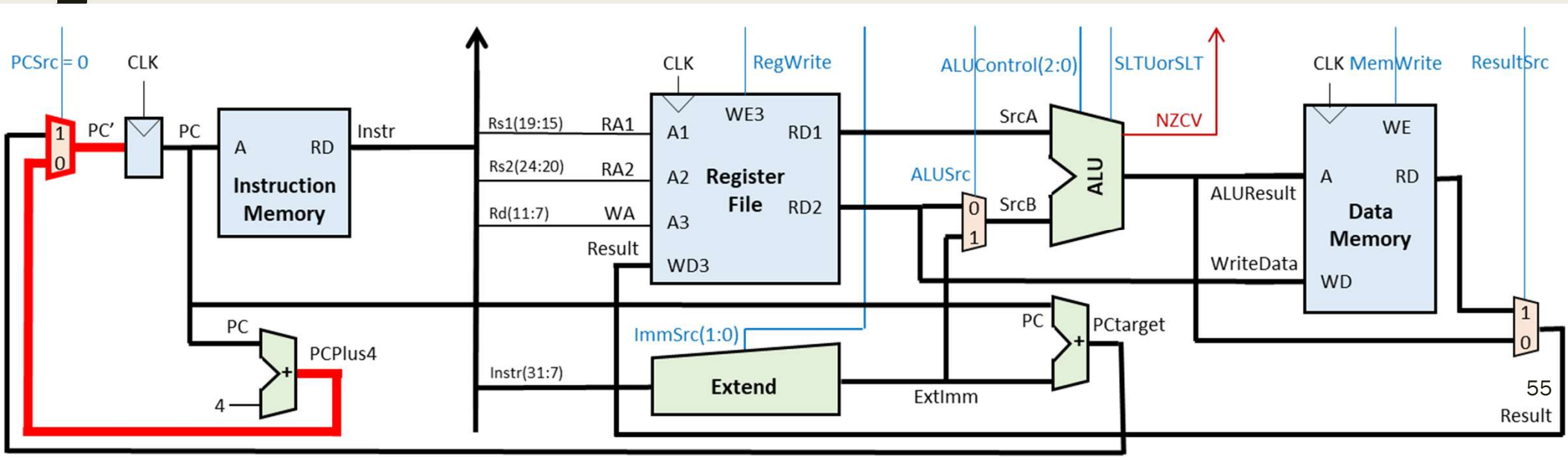
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
  - *σύνδεση του πεδίου Rd της εντολής (Instr11:7) στην είσοδο διευθύνσεων εγγραφής A3*
  - *σύνδεση της εξόδου ALUResult της μονάδας ALU με τη θύρα εγγραφής WD3 του αρχείου καταχωρητών 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

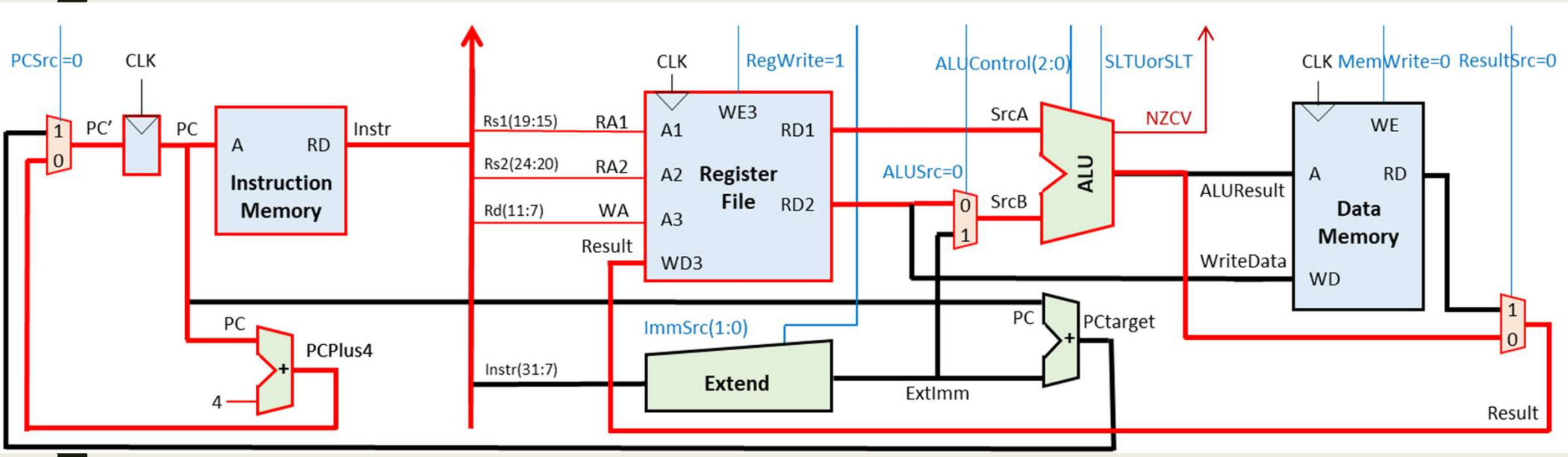
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- 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

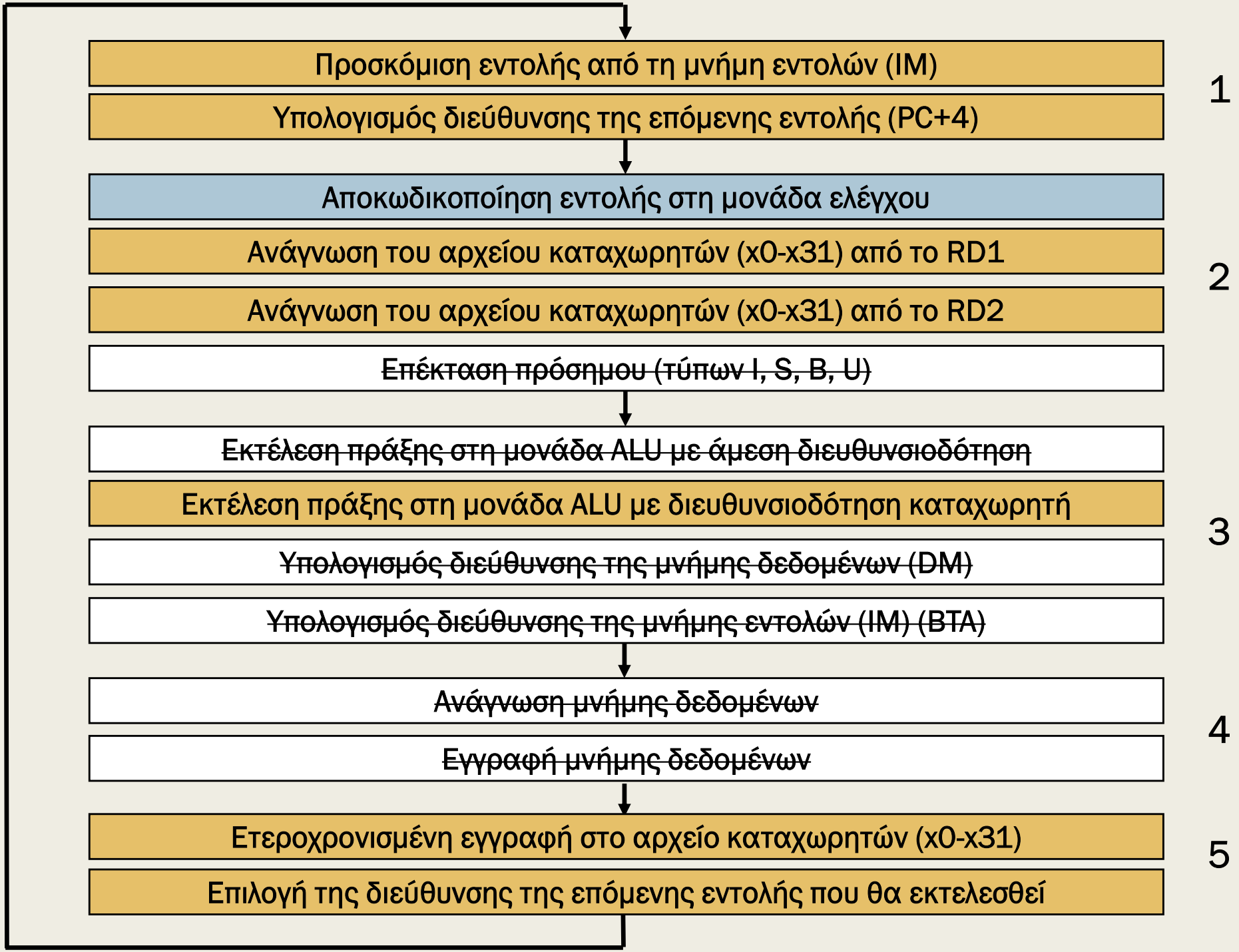


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALU

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

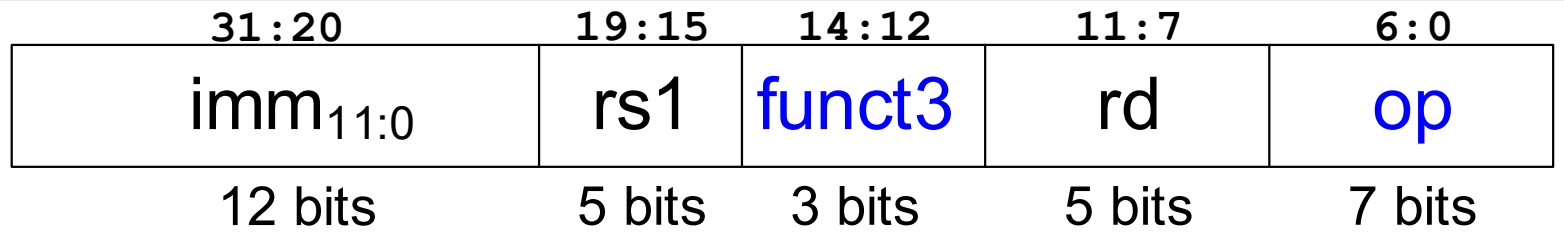


Λειτουργίες εντολών ALU (8)



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 and S_Ext(imm)`**
  - **`ORI   Rd, Rs1, imm  Rd = Rs1 or  S_Ext(imm)`**
  - **`XORI  Rd, Rs1, imm  Rd = Rs1 xor S_Ext(imm)`**
- Στη μορφή των εντολών 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 του άμεσου τελεστέου



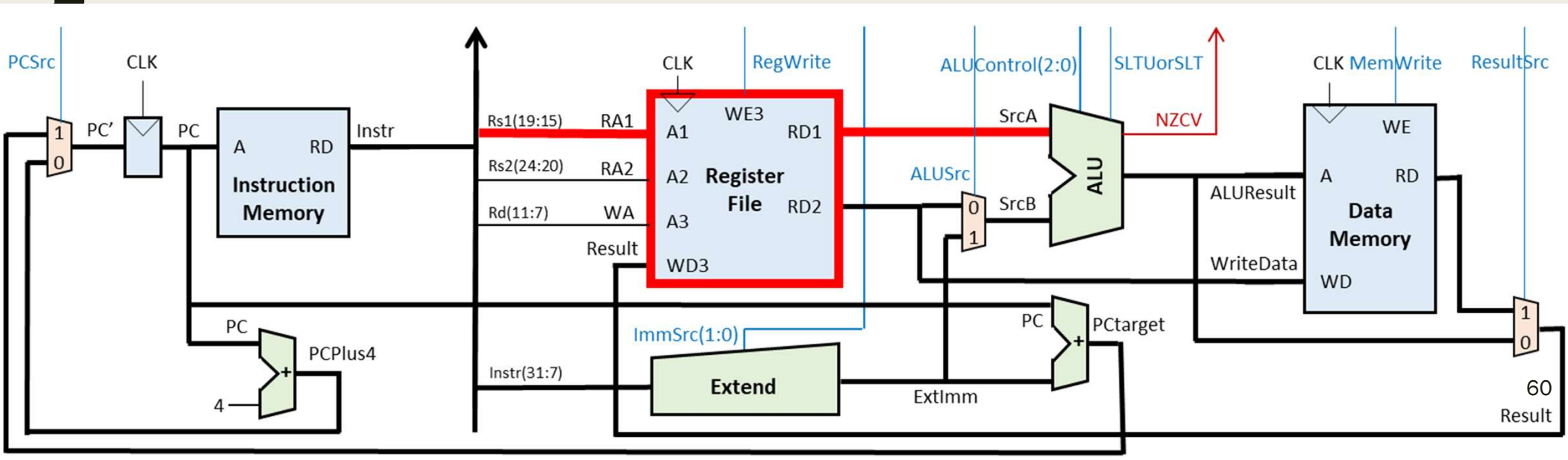
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

- 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

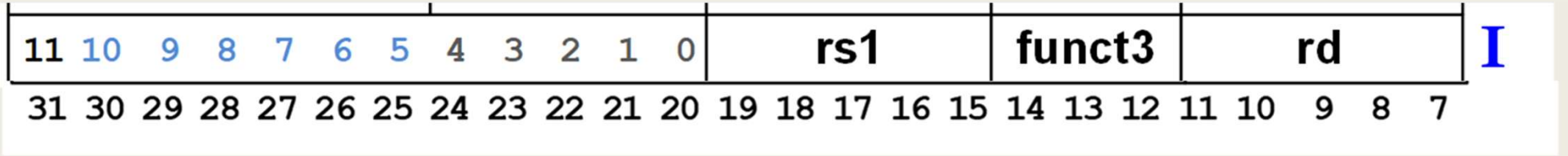
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD1

- ο τελεστέος 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)

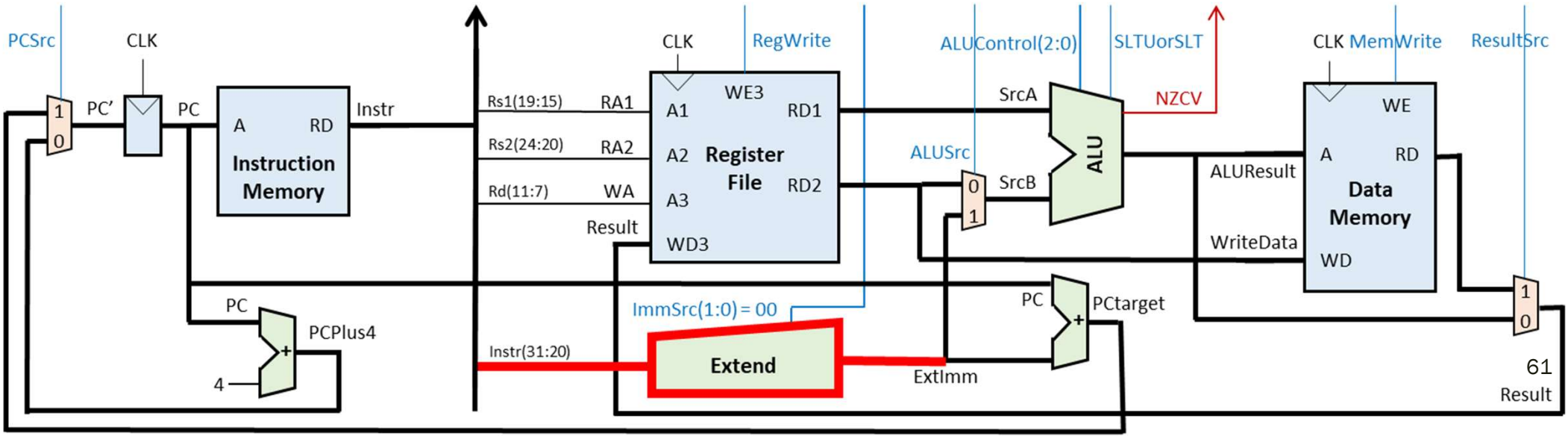


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

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



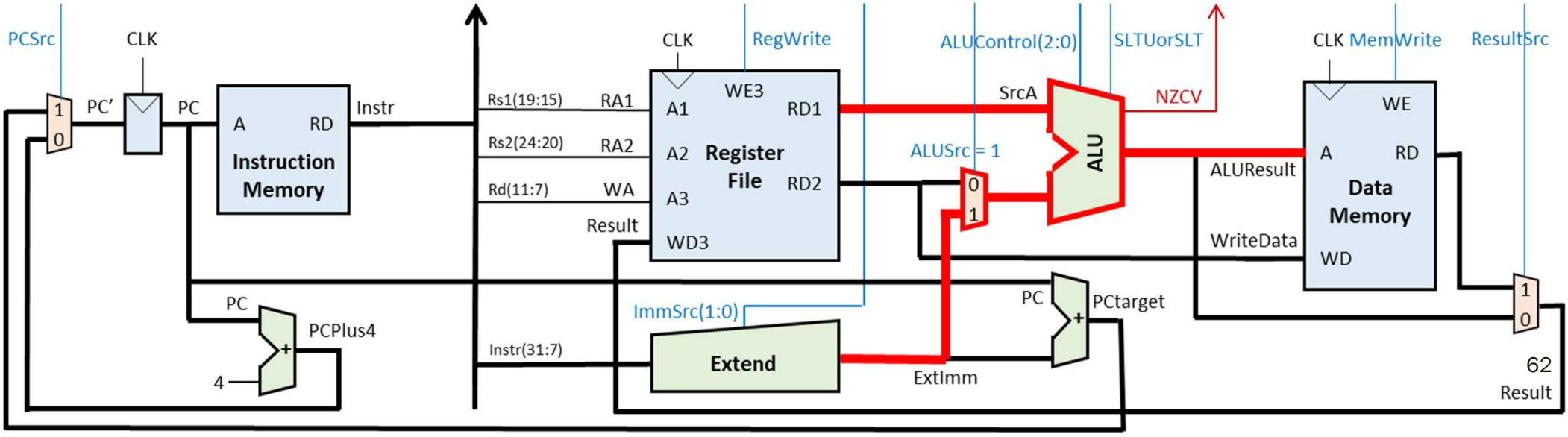
- Η εντολές ALUI διαθέτουν έναν προσημασμένο άμεσο τελεστέο imm(11:0) των 12 bit (instr(31:20)) που για να χρησιμοποιηθεί στις πράξεις πρέπει πρώτα να υποστεί επέκταση προσήμου από τα 12 bit στα 32 bit στην μονάδα Extend
  - **`Extend(31:0) = Sign_Ext(instr(31:20))`**
  - *Για επέκταση προσήμου τύπου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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

Εκτέλεση πράξης στη μονάδα ALU με άμεση διευθυνσιοδότηση

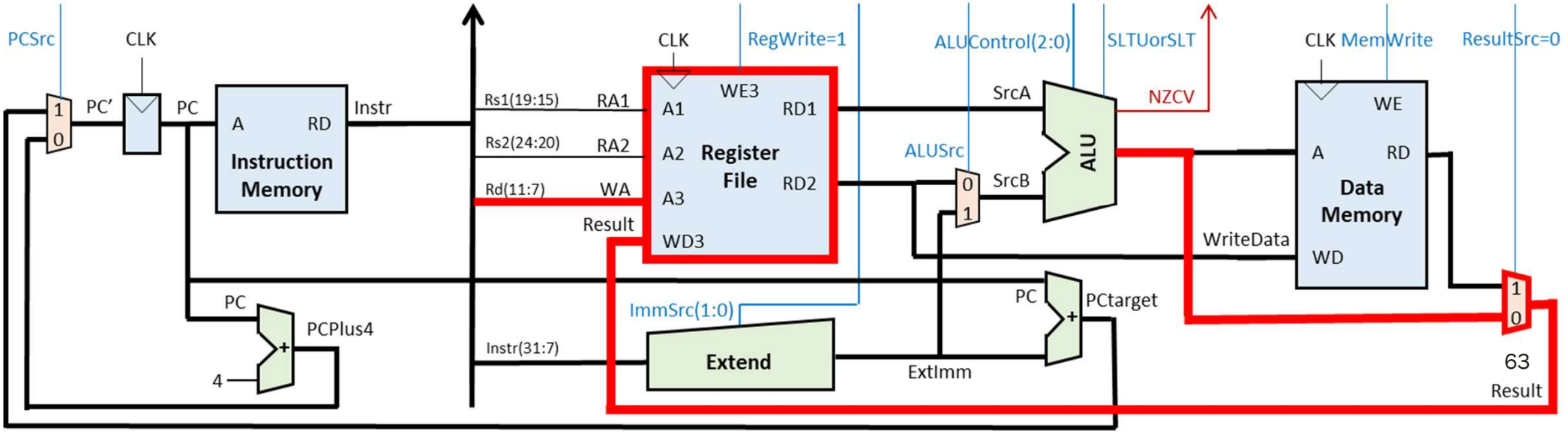
- Η μονάδα ALU εκτελεί τις αριθμητικές πράξεις της πρόσθεσης και της αφαίρεσης, καθώς και τις λογικές πράξεις AND, OR και XOR τύπου I μετά από:
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA των 32 bit της μονάδας ALU*
  - *σύνδεση της εξόδου ExtImm της μονάδας Extend με την είσοδο 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)

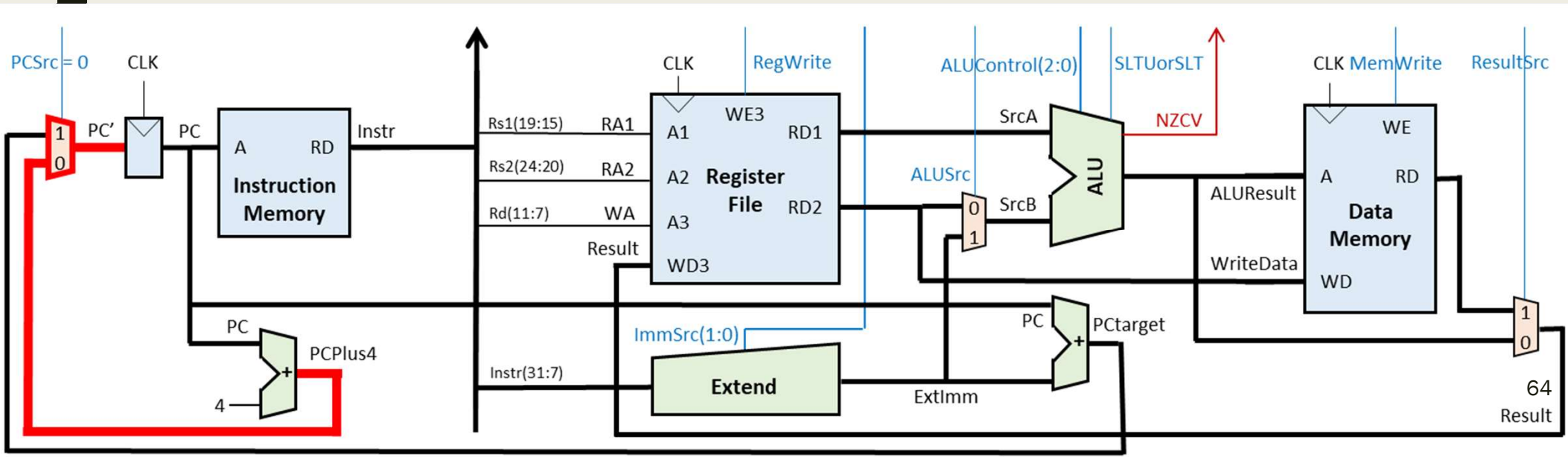
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
  - *σύνδεση του πεδίου Rd της εντολής (Instr11:7) στην είσοδο διευθύνσεων εγγραφής A3*
  - *σύνδεση της εξόδου ALUResult της μονάδας ALU με τη θύρα εγγραφής WD3 του αρχείου καταχωρητών 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

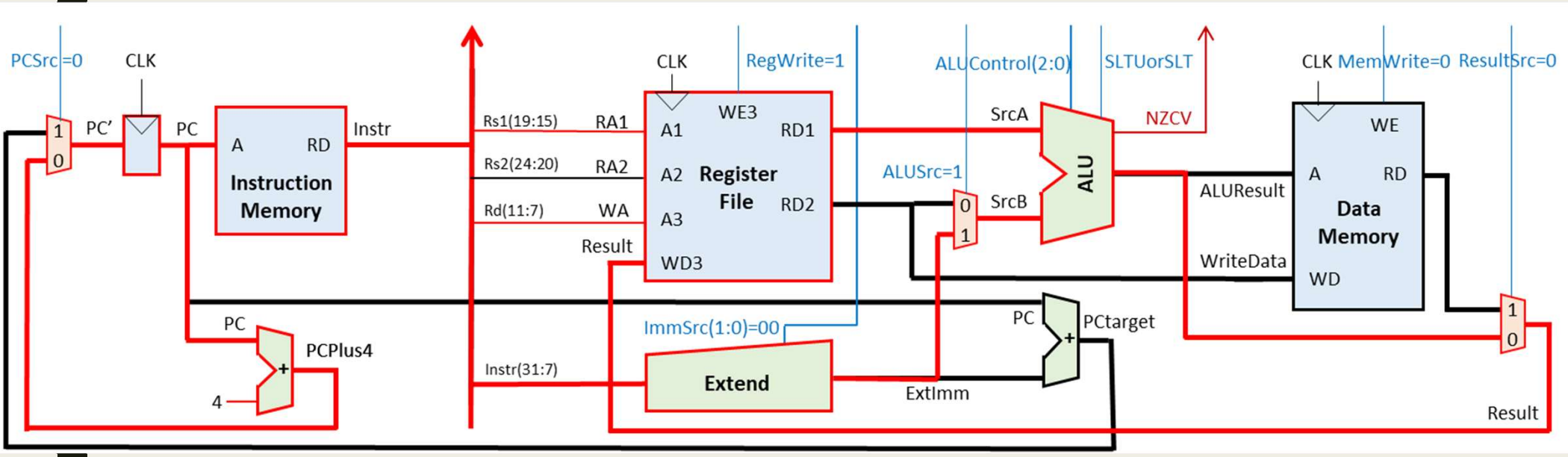
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- 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

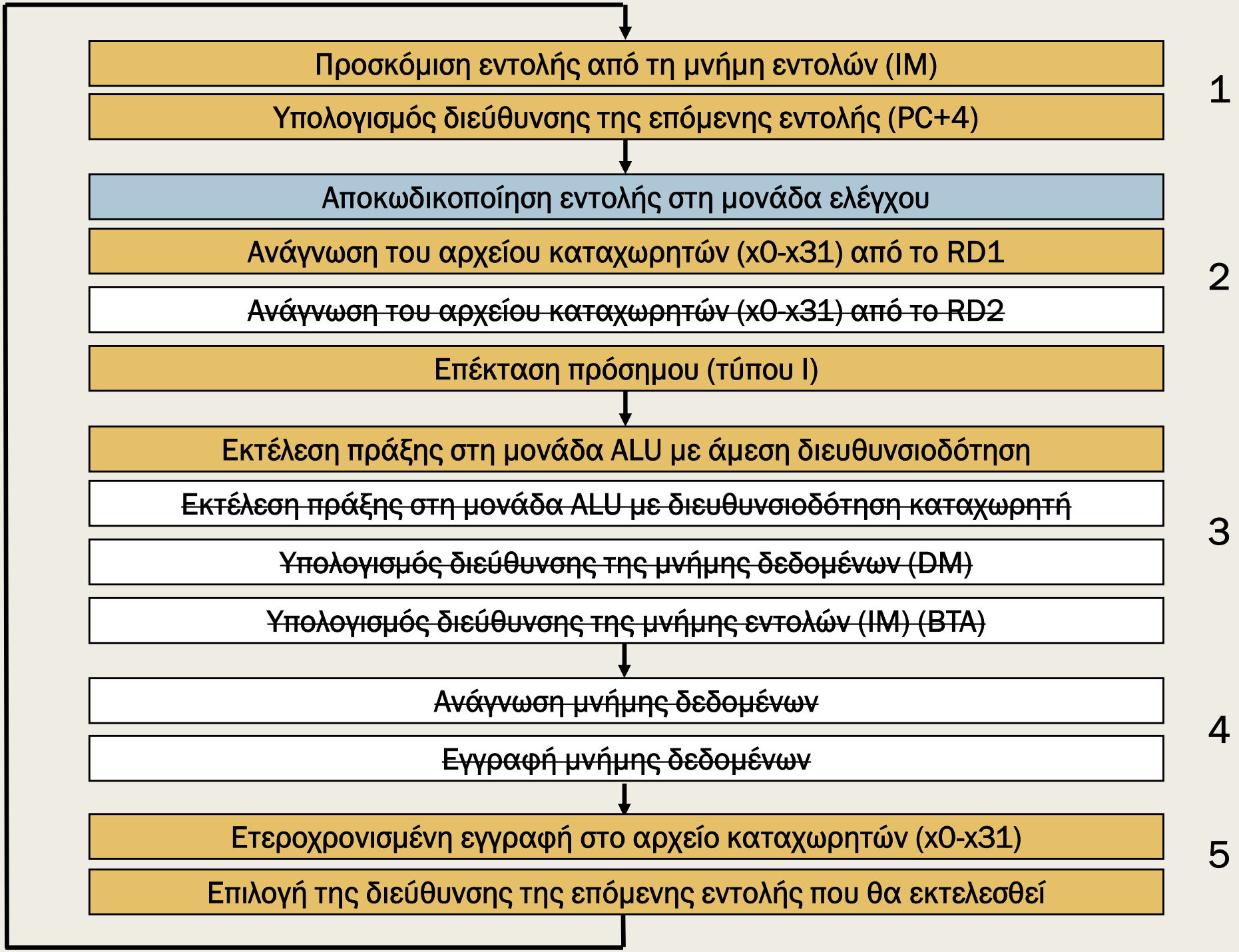


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές ALUI

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:

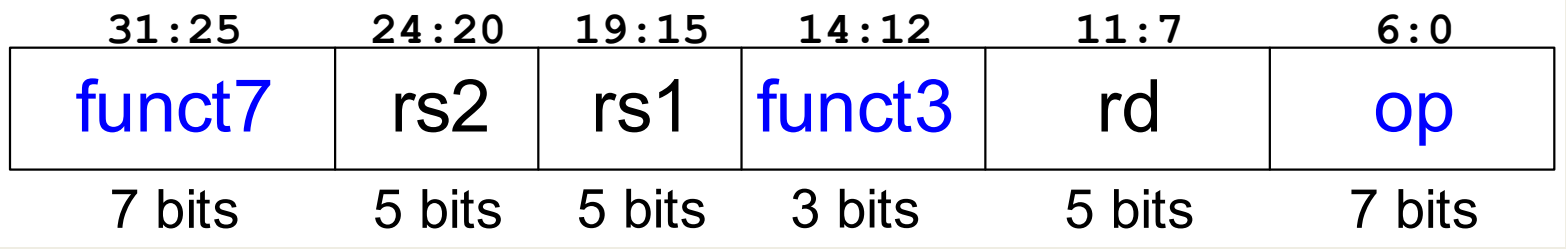


Λειτουργίες εντολών ALUI (8)



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 ⊕ V = 1, else 0 (N ⊕ V = 0)`**
  - **`SLTI: Rd = 1 if C = 0, else 0 (C = 1)`**

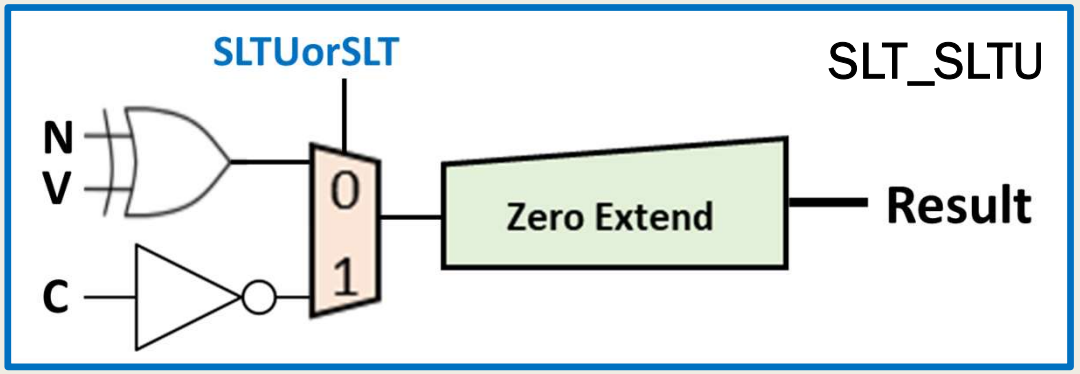


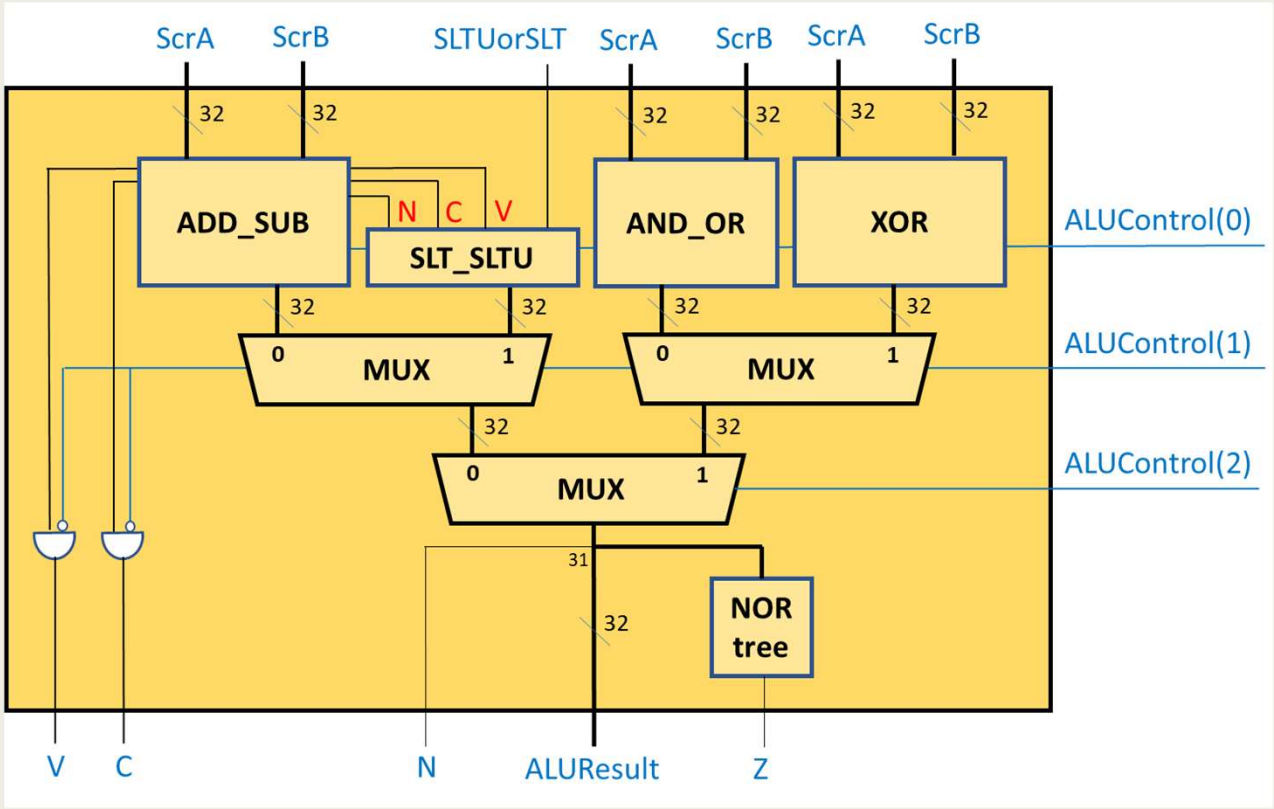
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

- 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 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 υπομονάδα SLT_SLTU της ALU πρέπει να υποστηρίζει την παραγωγή του*
    - Result = N xor V για την εντολή SLT (SLTUorSLT=0, ALUControl(2:0)=011), και
    - Result = 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

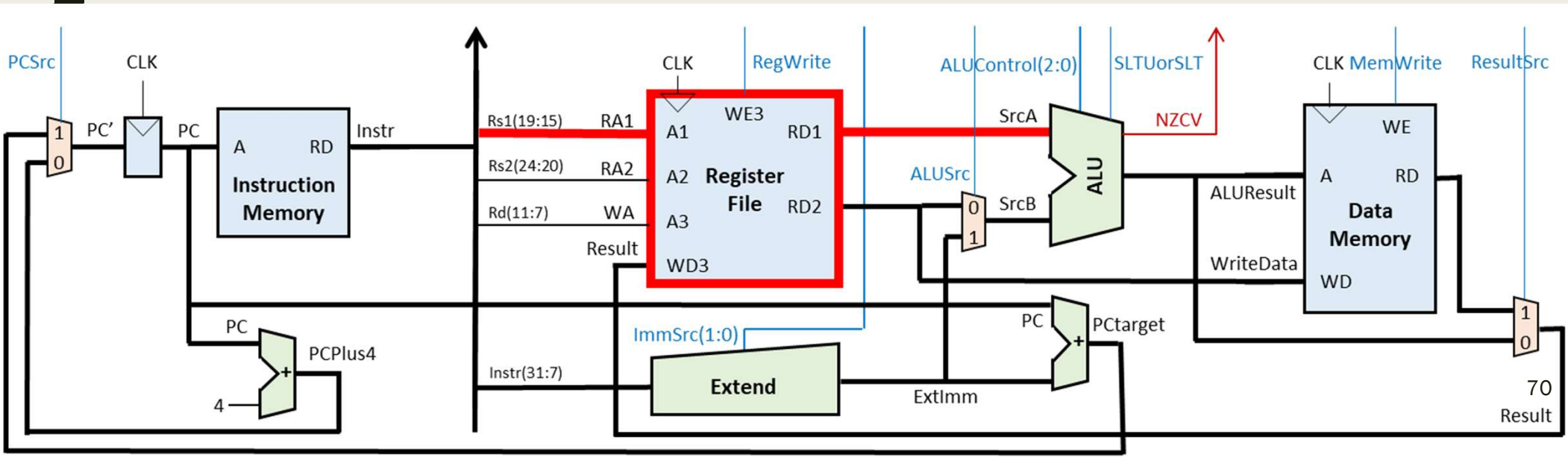




Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

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

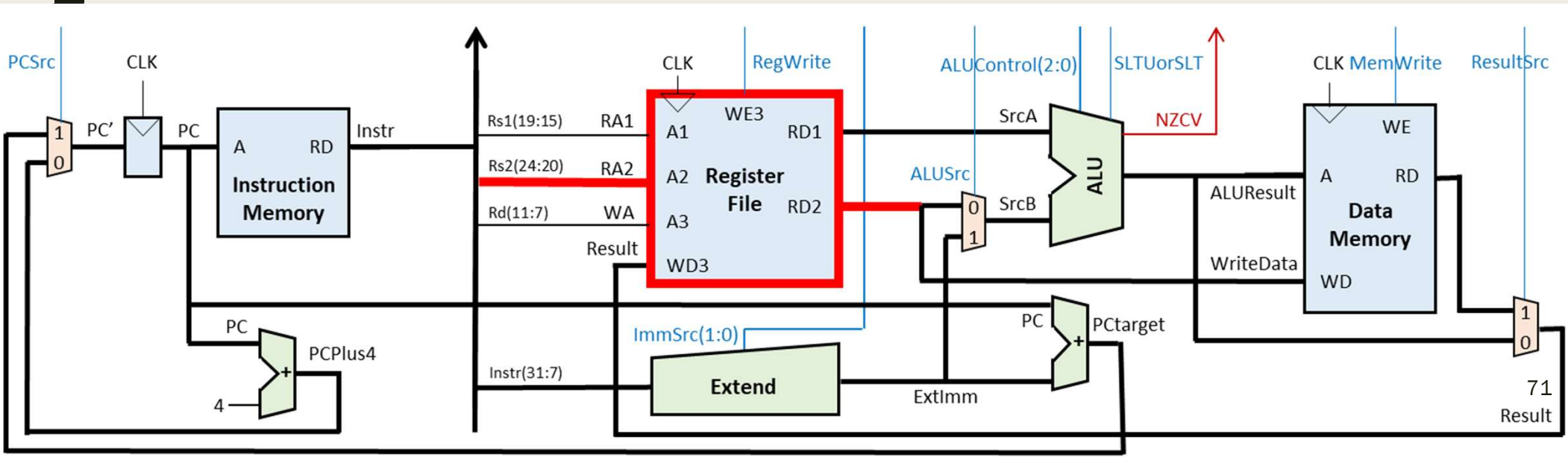
- ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs1 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
  - *σύνδεση του πεδίου Rs1 της εντολής (Instr19:15) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης RD1*
  - *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

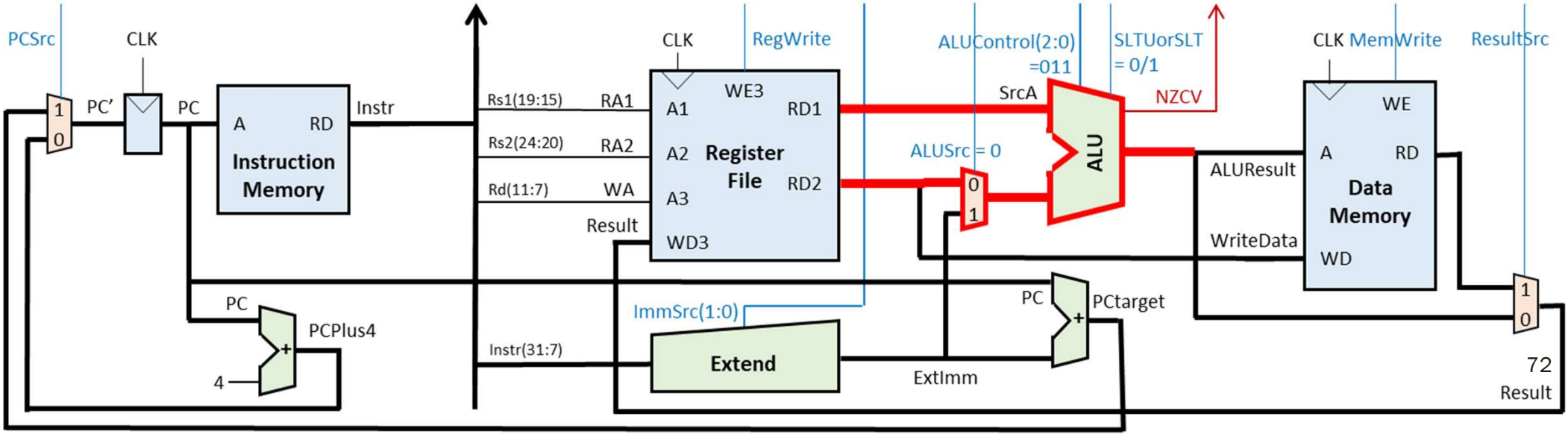
- ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs2 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD2
  - *σύνδεση του πεδίου Rs2 της εντολής (Instr24:20) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης RD2*
  - *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

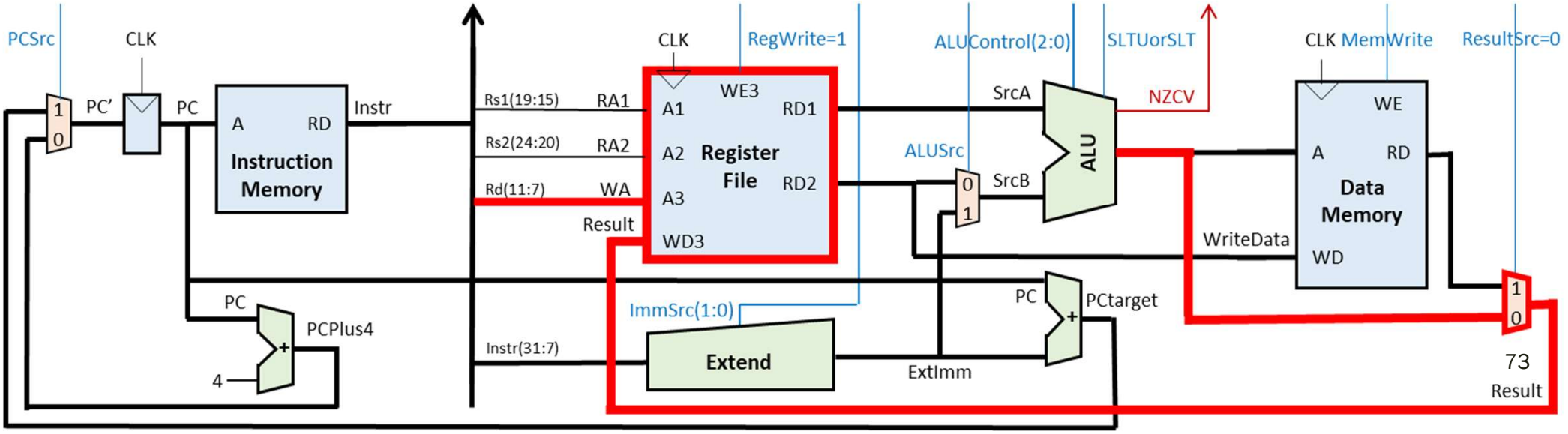
- Η μονάδα ALU εκτελεί αρχικά την πράξη της αφαίρεσης (με ALUControl(0)=1) μετά από:
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA της μονάδας ALU*
  - *σύνδεση της θύρας RD2 του αρχείου καταχωρητών με την είσοδο SrcB της μονάδας ALU*
- Τα παραγόμενα μετά την αφαίρεση ALU Flags (NCV) στην έξοδο του αθροιστή/αφαιρέτη χρησιμοποιούνται για να παραχθεί η έξοδος Result ως εξής:
  - **Result = N xor V** για την εντολή SLT (SLTUorSLT=0, ALUControl(2:0)=011), και
  - **Result = 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)

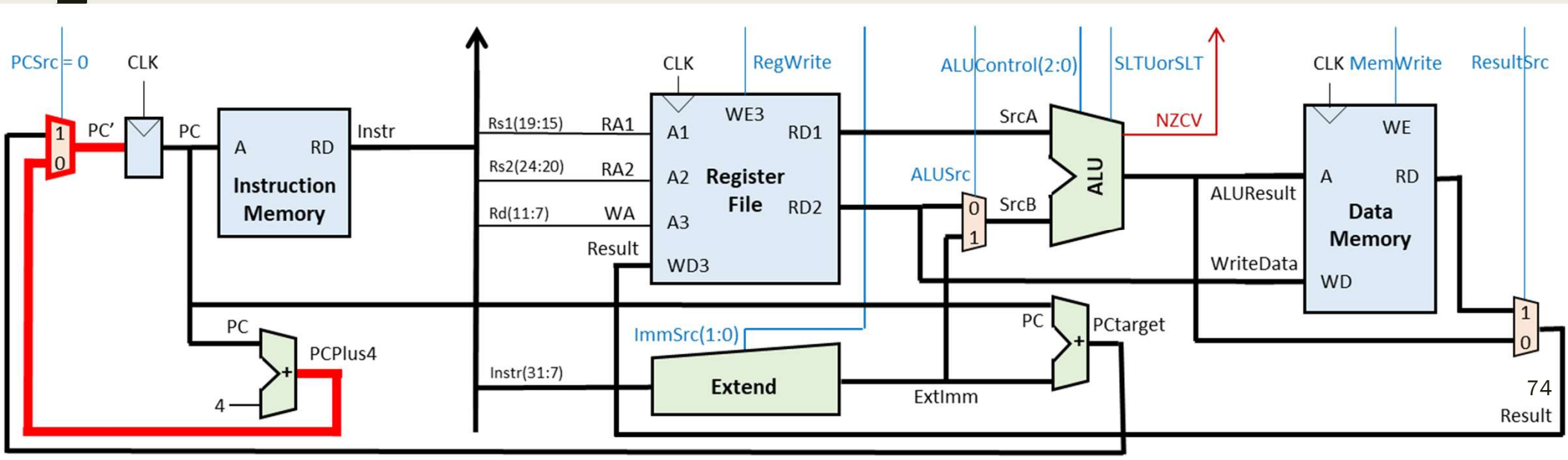
- Το αποτέλεσμα της πράξης στη μονάδα ALU (ALUResult) αποθηκεύεται στον καταχωρητή προορισμού Rd του αρχείου καταχωρητών
  - *σύνδεση του πεδίου Rd της εντολής (Instr11:7) στην είσοδο διευθύνσεων εγγραφής A3*
  - *σύνδεση της εξόδου ALUResult της μονάδας ALU με τη θύρα εγγραφής WD3 του αρχείου καταχωρητών 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

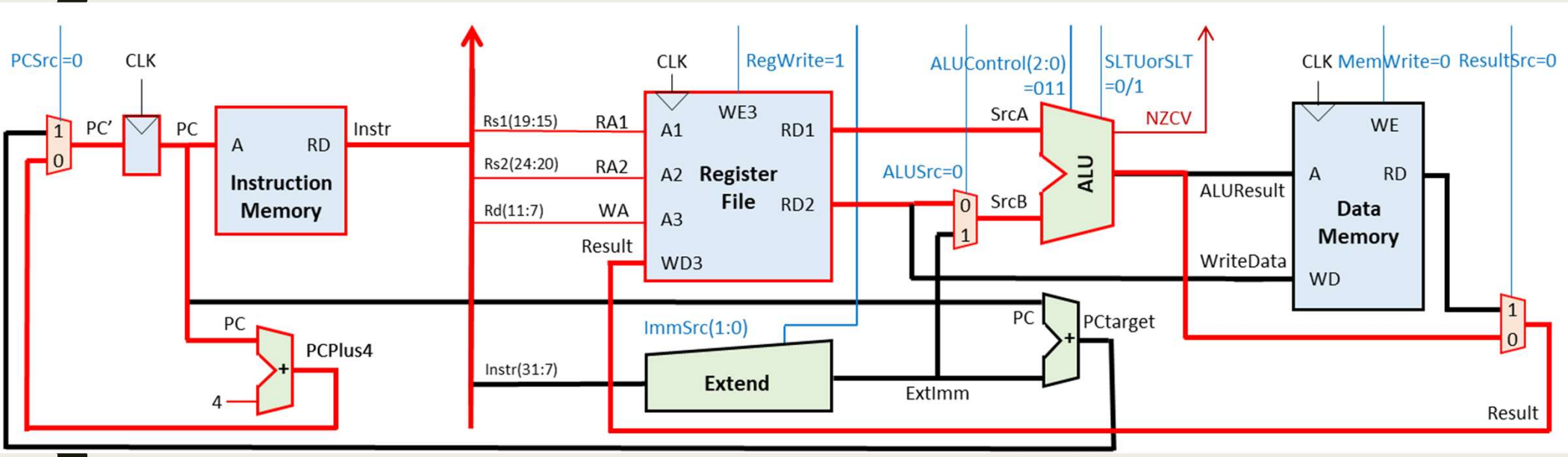
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- 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

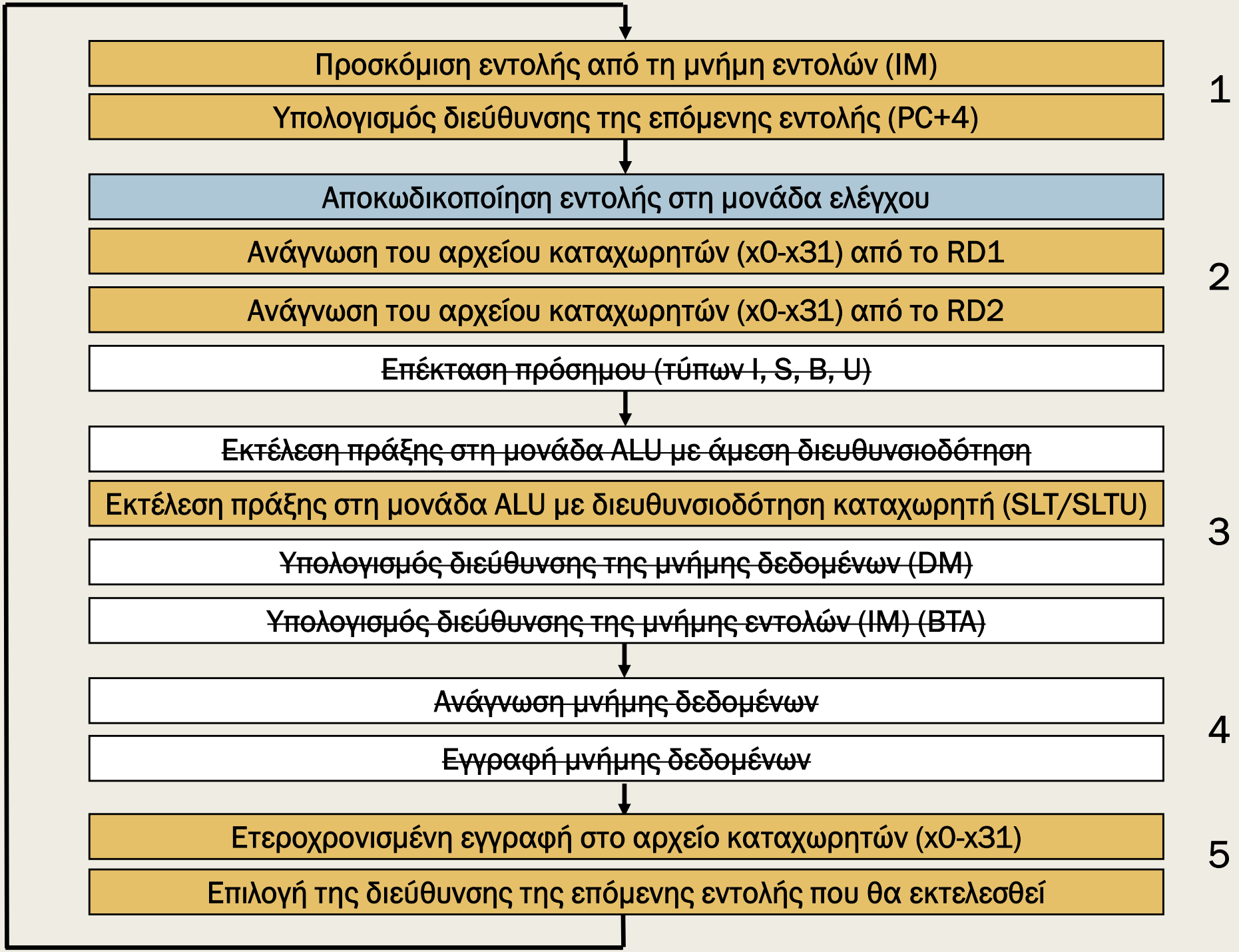


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές SLT, SLTU

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



Λειτουργίες εντολών SLT/SLTU (8)

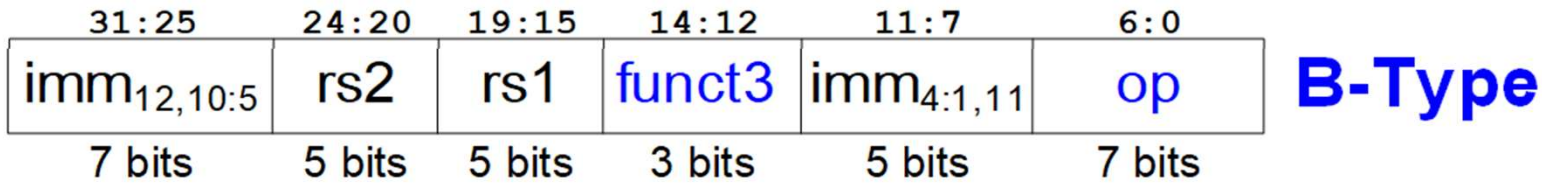


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- 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(31:25)&instr(11:7))*

`BTA = PC + S_Ext (imm(12:1)x2)`

- 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*



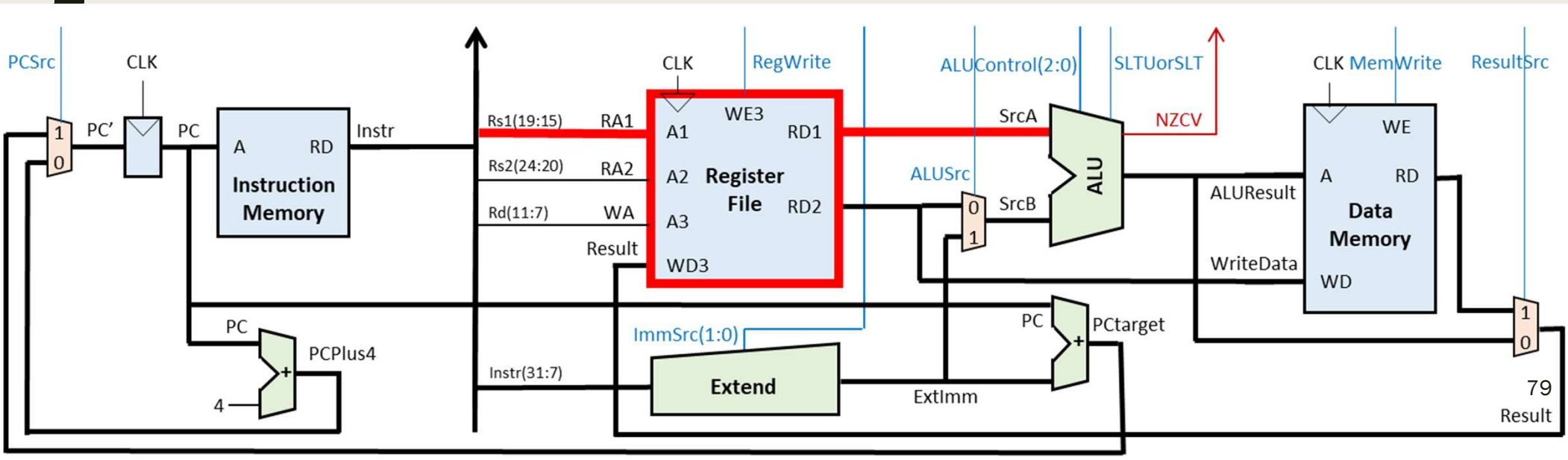
Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- Απαιτούμενες λειτουργίες των εντολών Branch που ήδη υποστηρίζονται:
  - *ο 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*
  - *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 τελεστέου 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

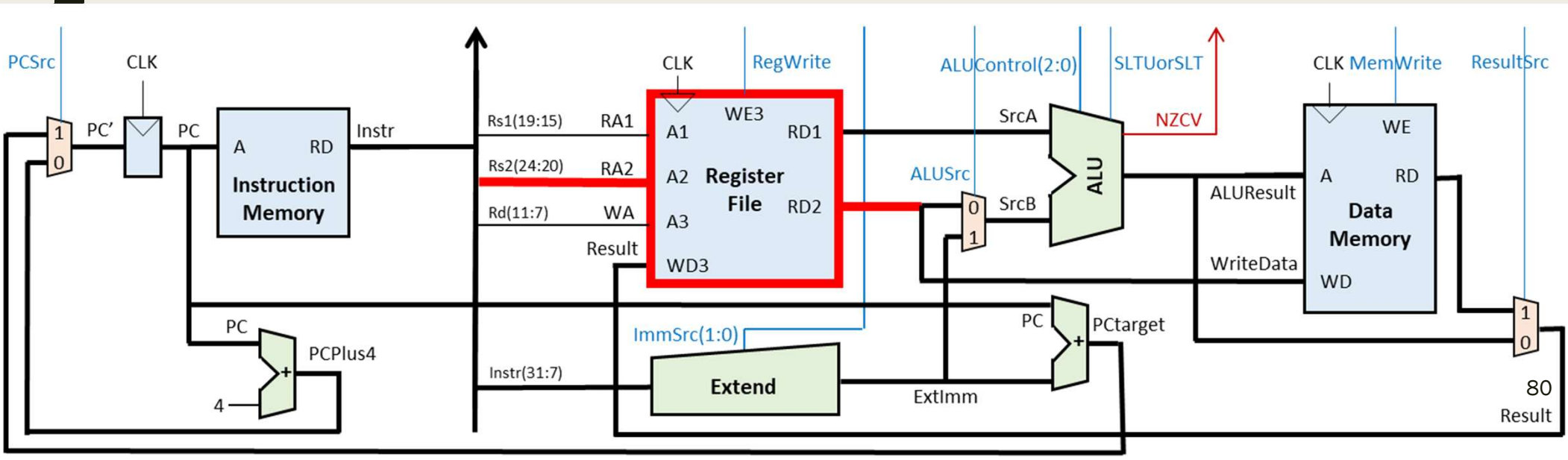
- ο τελεστέος που είναι αποθηκευμένος στον καταχωρητή προέλευσης Rs1 του αρχείου καταχωρητών (x0-x31) διαβάζεται στη θύρα ανάγνωσης RD1
  - *σύνδεση του πεδίου Rs1 της εντολής (Instr19:15) στην είσοδο διευθύνσεων ανάγνωσης A1*
  - *ασύγχρονη ανάγνωση περιεχομένου καταχωρητή από τη θύρα ανάγνωσης RD1*
  - *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

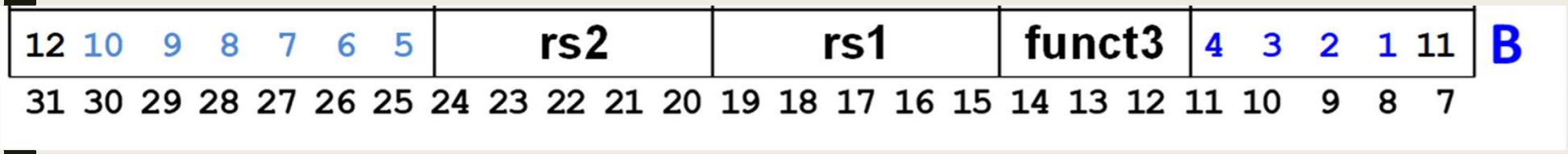
Ανάγνωση του αρχείου καταχωρητών (x0-x31) από το RD2

- ο τελεστέος 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)

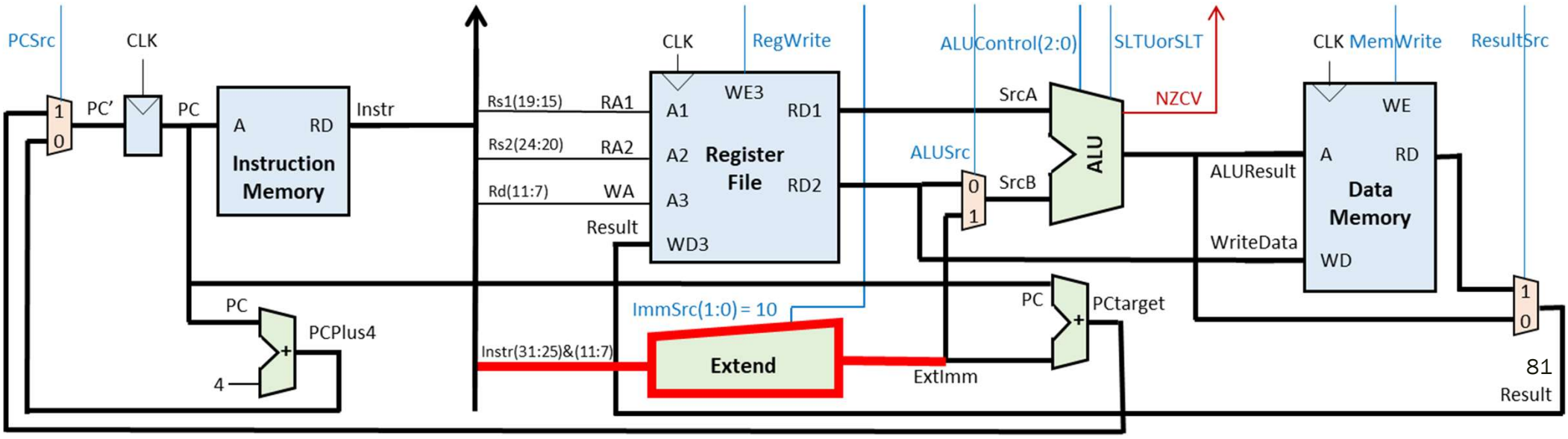


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

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



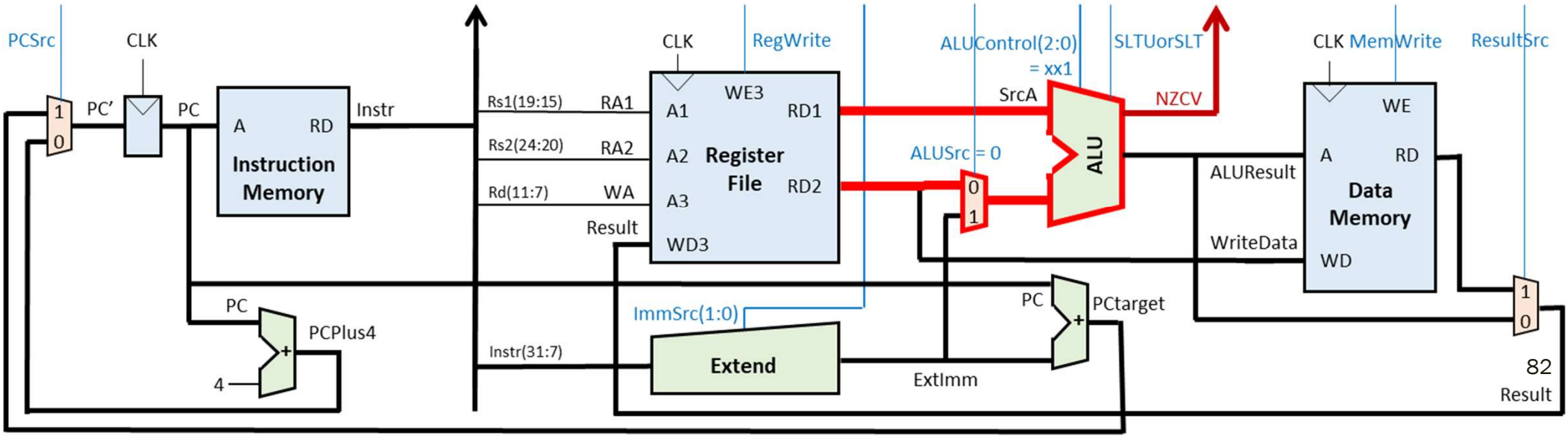
- Η εντολές Branch απαιτούν το υπολογισμό της Branch Target Address (BTA) που προκύπτει με επέκταση προσήμου από τα 13 bit στα 32 bit στην μονάδα **Extend**
  - **`Extend(31:0) = Sign_Ext(instr(31)& instr(7)&instr(30:25)&instr(11:8)&'0')`**
  - *Για επέκταση προσήμου τύπου B τα σήματα ελέγχου είναι:*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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

Εκτέλεση πράξης στη μονάδα ALU με διευθυνσιοδότηση καταχωρητή

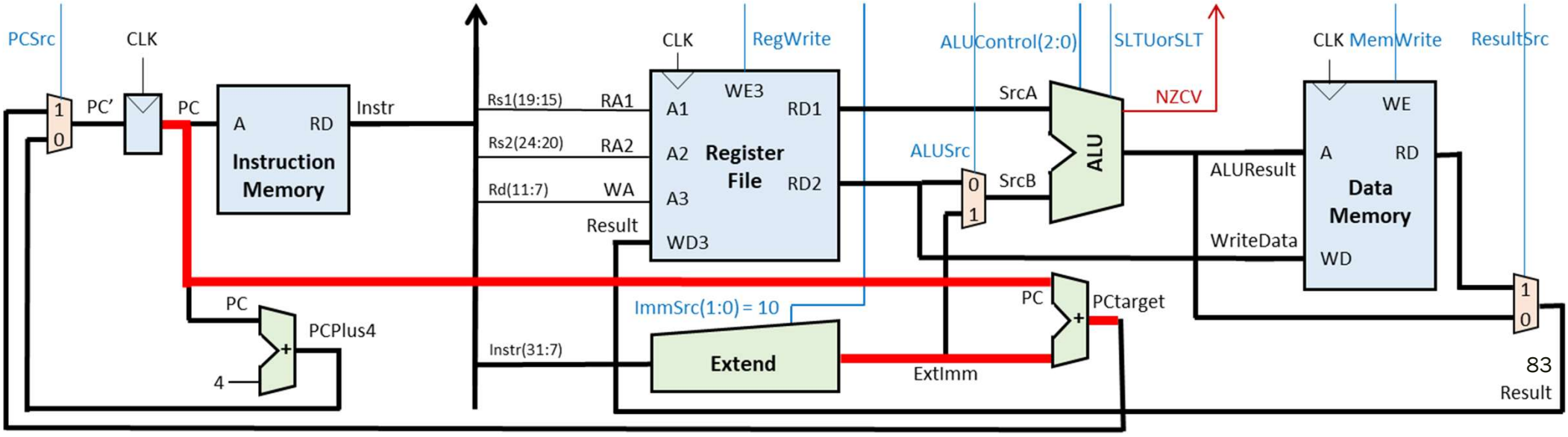
- Η μονάδα ALU εκτελεί αρχικά την πράξη της αφαίρεσης (με ALUControl(0)=1) μετά από:
  - *σύνδεση της θύρας RD1 του αρχείου καταχωρητών με την είσοδο SrcA της μονάδας ALU*
  - *σύνδεση της θύρας RD2 του αρχείου καταχωρητών με την είσοδο SrcB της μονάδας ALU*
- Τα παραγόμενα μετά την αφαίρεση ALU Flags (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 κατάληλο 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)

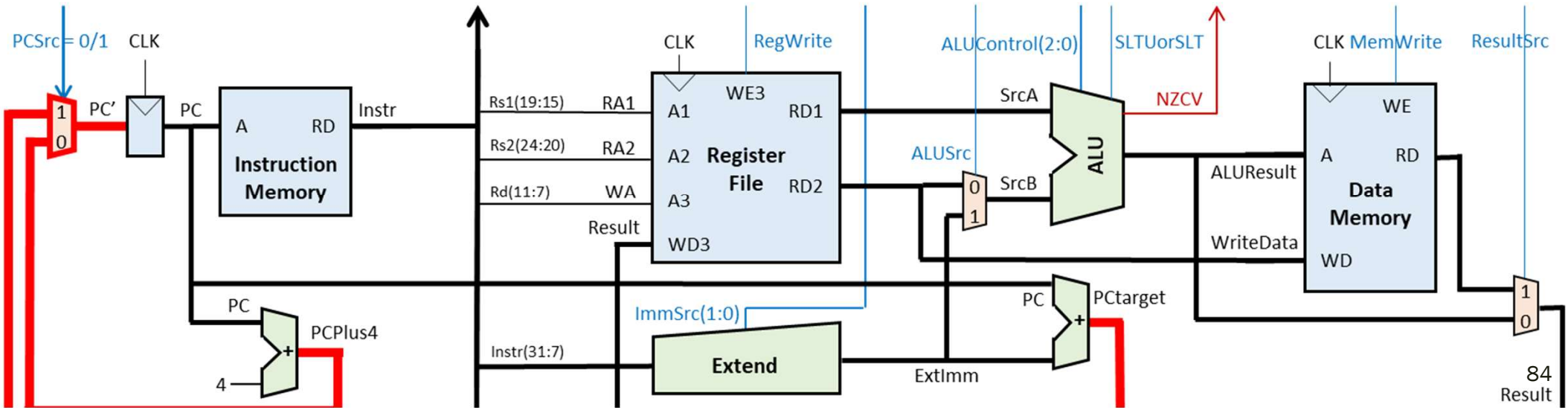
- Σε ένα διακριτό αθροιστή υπολογίζεται η διεύθυνση BTA της μνήμης εντολών (IM)
  - *σύνδεση της εξόδου PC του μετρητή προγράμματος με την πρώτη είσοδο του διακριτού αθροιστή*
  - *σύνδεση της εξόδου ExtImm της μονάδας Extend με τη δεύτερη είσοδο του διακριτού αθροιστή*
  - *εκτελείται στον διακριτό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



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολή Branch

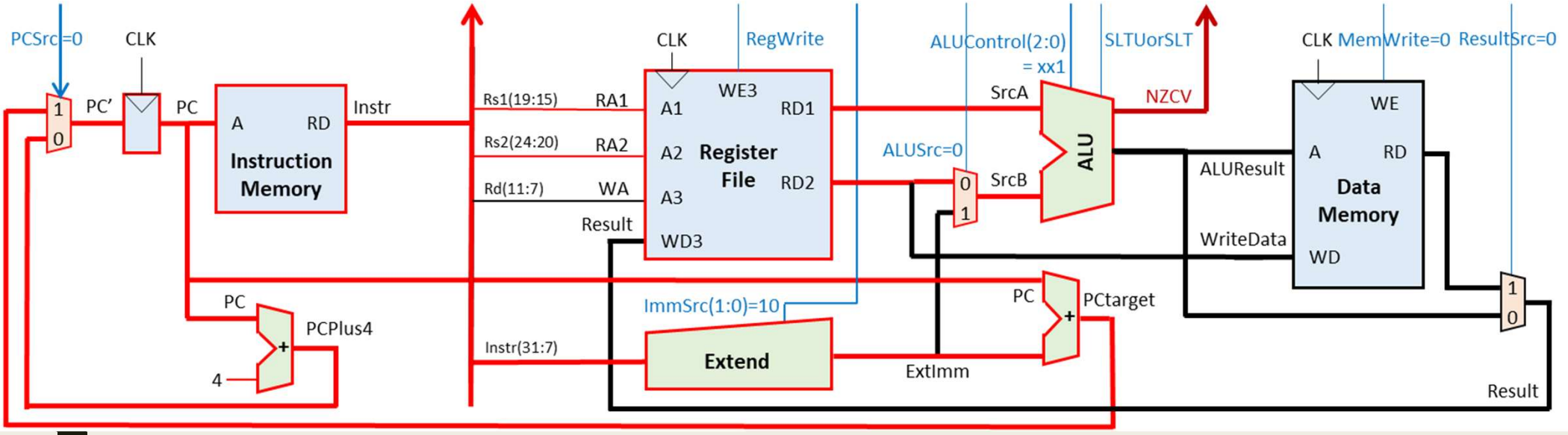
Επιλογή της διεύθυνσης της επόμενης εντολής που θα εκτελεσθεί

- Ο πολυπλέκτης επιλογής διεύθυνσης επόμενης εντολής επιλέγει τη διεύθυνση:
  - PC' = PCtarget (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

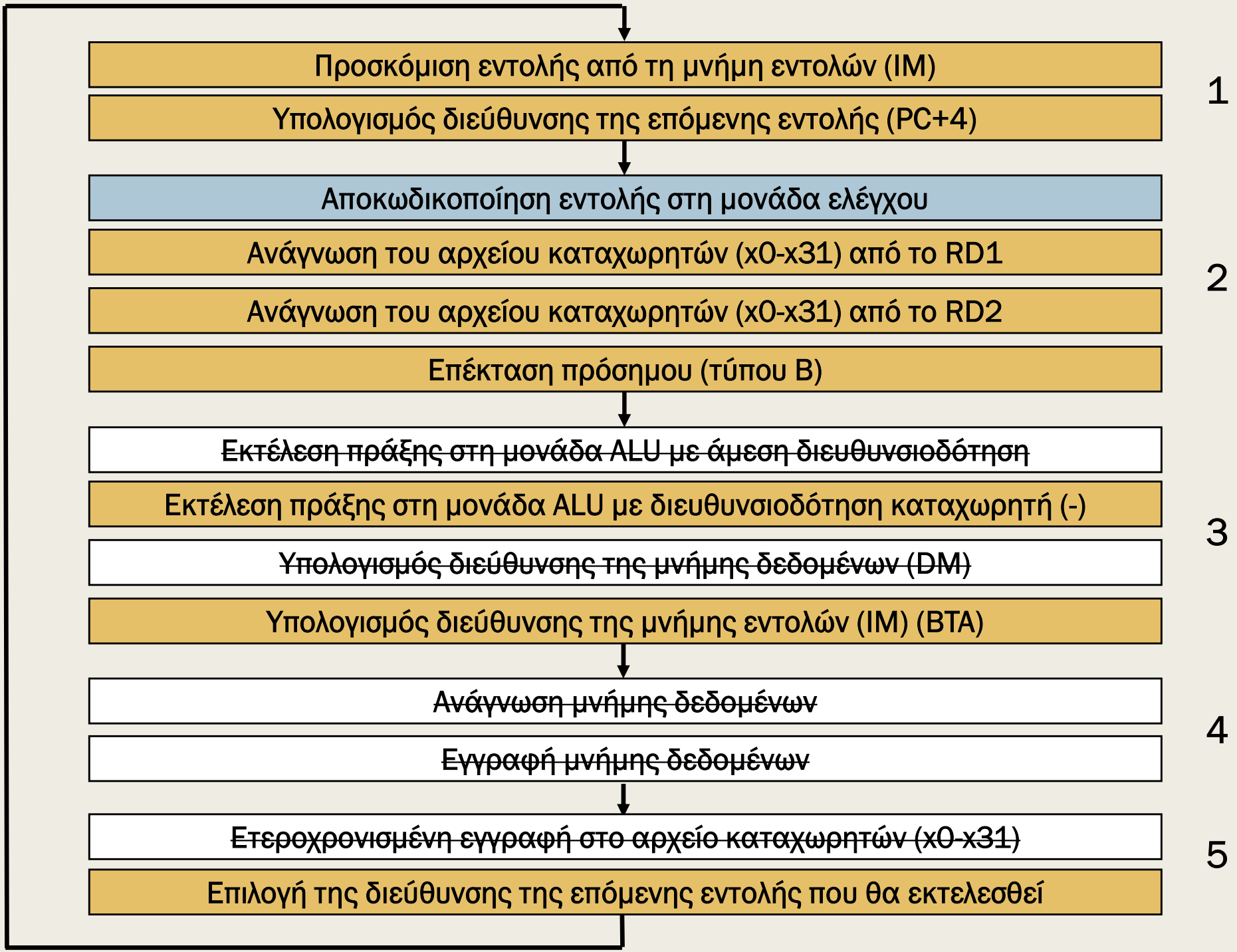


Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Εντολές Branch

- Συνολική ροή δεδομένων και τιμές στα σήματα ελέγχου:



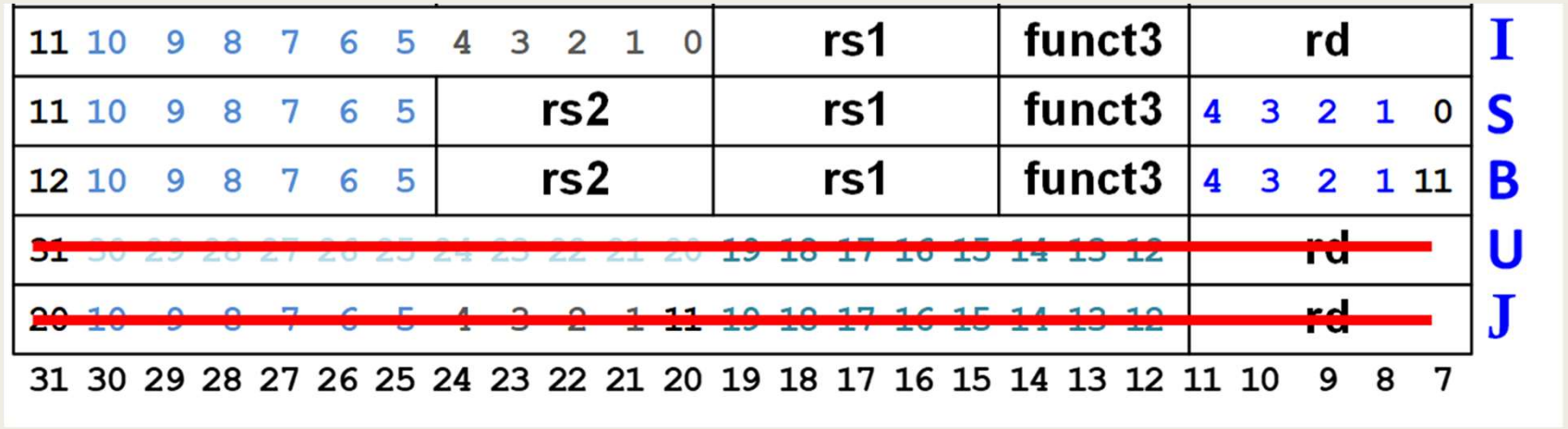
Λειτουργίες εντολών Branch (9)



Επεξεργαστής ενός κύκλου: μελέτη διαδρομής δεδομένων – Μονάδα Extend

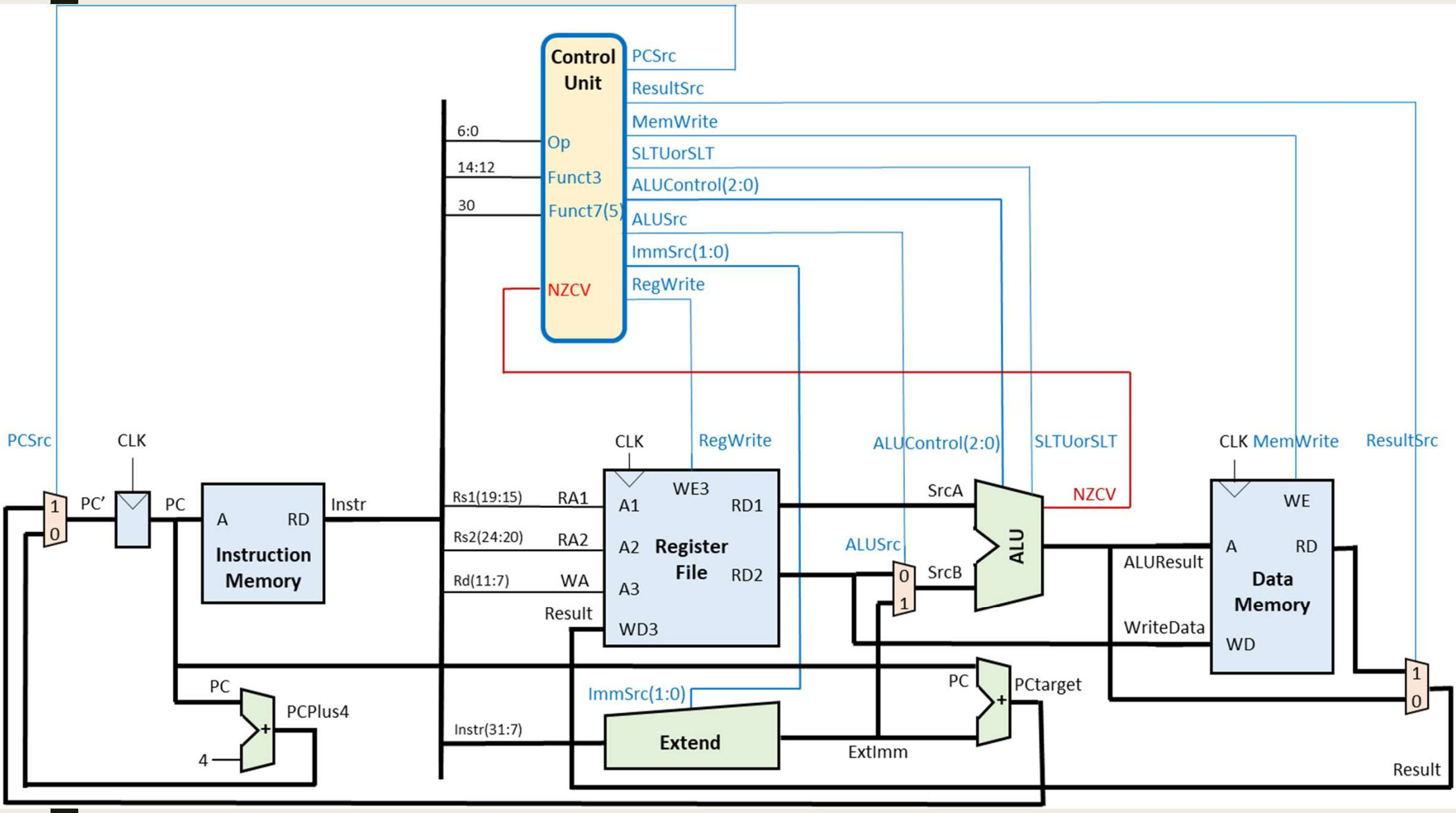
- Για το βασικό σύνολο εντολών της αρχιτεκτονικής RISC-V, που έχουμε μελετήσει μέχρι τώρα, η μονάδα Extend εκτελεί τις ακόλουθες διακριτές επεκτάσεις προσήμου στα 32 bit:

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



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) = 01000 (στην εντολή SW)
  - op(6:2) = 01100 (σε όλες τις εντολές ALU τύπου R)
  - op(6:2) = 00100 (σε όλες τις εντολές ALU τύπου I)
  - op(6:2) = 11000 (σε όλες τις εντολές Branch)
  - op(6:2) = 01101 (στην εντολή LUI)
  - 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
		LUI	0110111
		JALR	1100111

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

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

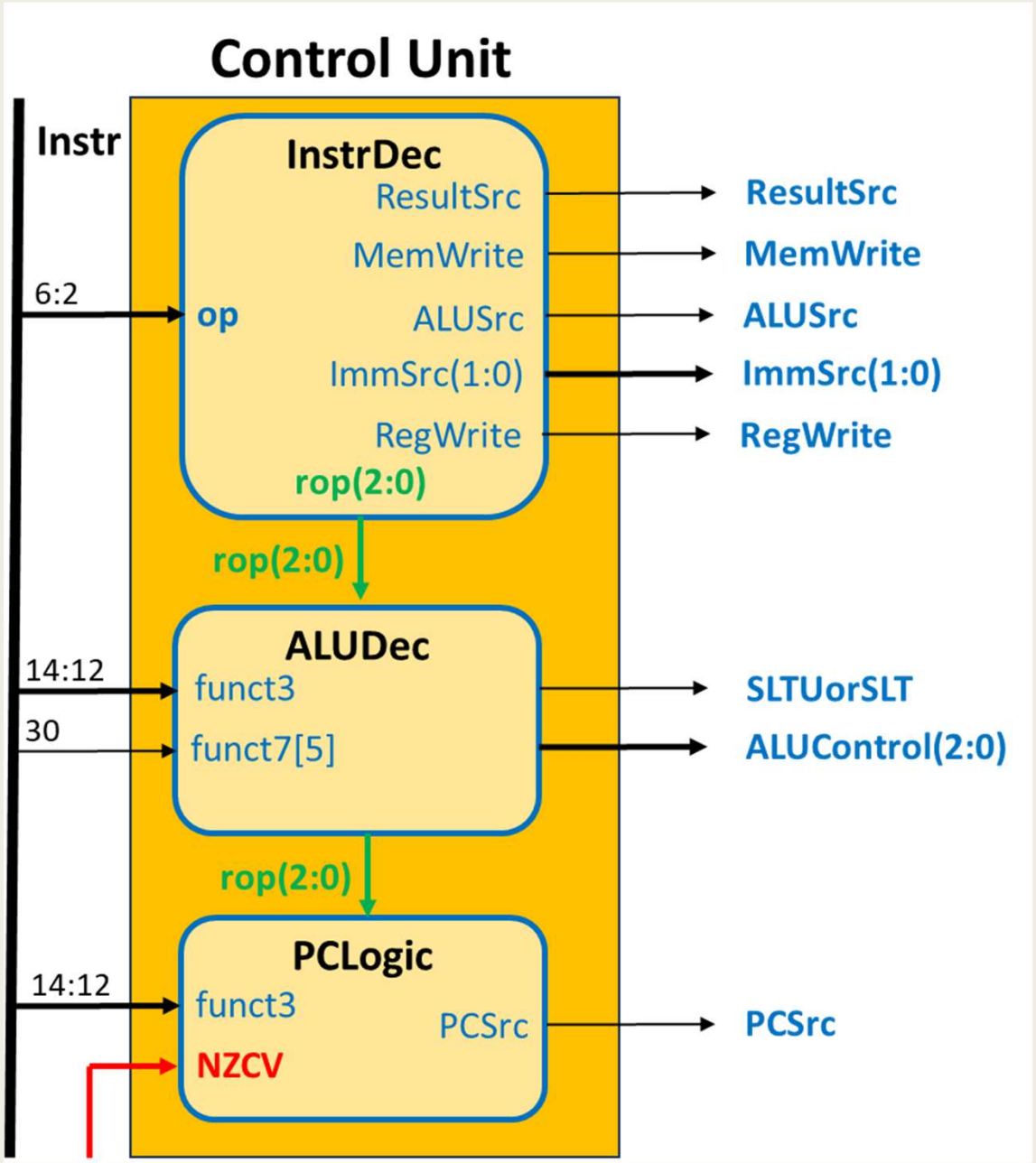
Οι εντολές που έχουν κοινό rop(2,0) ξεχωρίζουν με βάση τα πεδία funct3 και funct7(5)

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

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

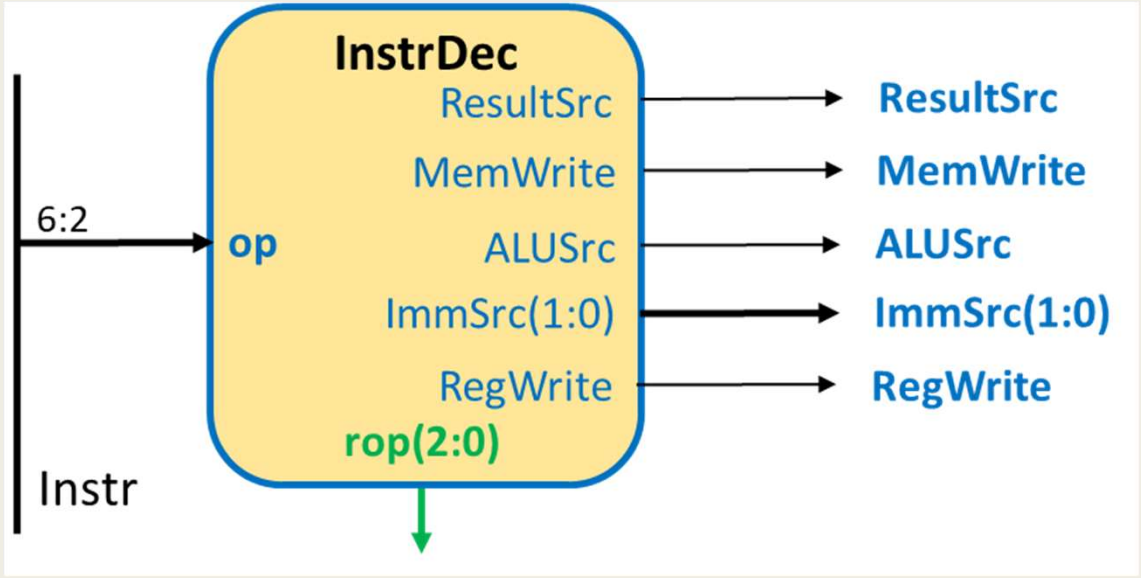
1. Κύριος αποκωδικοποιητής εντολής (InstrDec)
   - *η υπομονάδα που αποκωδικοποιεί το πεδίο op της εντολής (Instr(6:2) και παράγει τα σήματα επιλογής πολυπλεκτών ALUSrc και ResultSrc, το σήμα επιλογής επέκτασης προσήμου ImmSrc(1:0), τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών RegWrite και στη μνήμη 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)=100, 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 (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. Αποκωδικοποιητής εντολής (InstrDec)

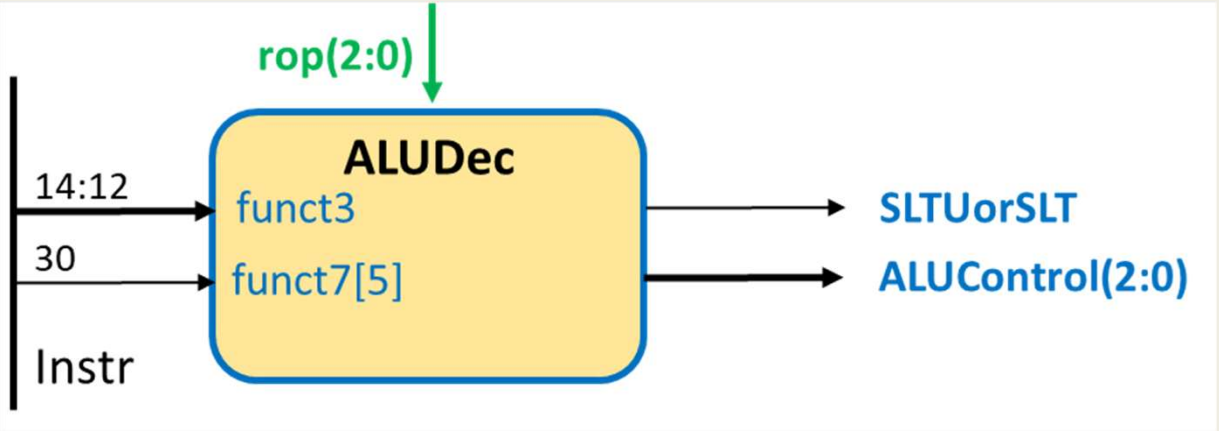
Εντολή	op(6:2)	rop(2:0)	Result Src	Mem Write	ALU Src	Imm Src(1:0)	Reg Write
LW	00000	000	1	0	1	00	1
SW	01000	001	X	1	1	01	0
ALU	01100	010	0	0	0	XX	1
ALUI	00100	011	0	0	1	00	1
Branch	11000	100	X	0	0	10	0
LUI	01101	101	TBD	TBD	TBD	TBD	TBD
JALR	11001	11X	TBD	TBD	TBD	TBD	TBD

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(6:0)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (2:0)	SLTUorSLT
ADD	0110011	010	000	0	add	000	X
SUB	0110011	010	000	1	sub	001	X
AND	0110011	010	111	0	and	100	X
OR	0110011	010	110	0	or	101	X
XOR	0110011	010	100	0	xol	11X	X
SLL	0110011	010	001	0	sll	TBD	X
SRL	0110011	010	101	0	srl	TBD	X
SRA	0110011	010	101	1	sra	TBD	X
SLT	0110011	010	010	0	slt	01X	0
SLTU	0110011	010	011	0	sltu	01X	1

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

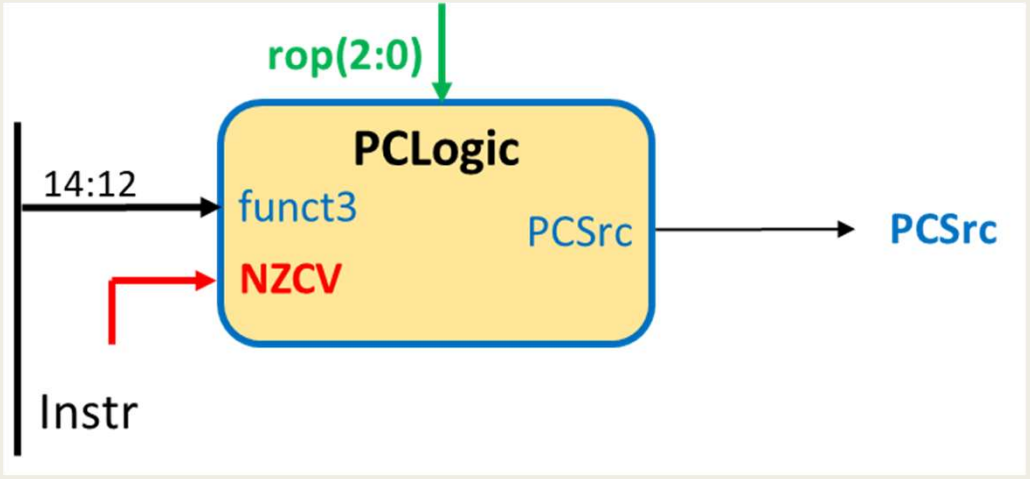
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:0)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (2:0)	SLTUorSLT
ADDI	0010011	011	000	X	add	000	X
ANDI	0010011	011	111	X	and	100	X
ORI	0010011	011	110	X	or	101	X
XORI	0010011	011	100	X	xol	11X	X
LW	0000011	000	010	X	add	000	X
SW	0100011	001	010	X	add	000	X
BEQ	1100011	100	000	X	sub	001	X
BNE	1100011	100	001	X	sub	001	X
LUI	0110111	101	XXX	X	-	XXX	X
JALR	1100111	11X	000	X	add	000	X

Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

- *η υπομονάδα λαμβάνει το εσωτερικό σήμα rop(2:0)=100, 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 PCSrch*



Επεξεργαστής ενός κύκλου: μελέτη μονάδας ελέγχου

3. Λογική επιλογής διεύθυνσης επόμενης εντολής (PCLogic)

Εντολή	op(6:0)	rop(2:0)	funct3	NZCV	PCSrc
BEQ	1100011	100	000	X0XX	0
BEQ	1100011	100	000	X1XX	1
BNE	1100011	100	001	X0XX	1
BNE	1100011	100	001	X1XX	0
LW	00000	000	XXX	XXXX	0
SW	01000	001	XXX	XXXX	0
ALU	01100	010	XXX	XXXX	0
ALUI	00100	011	XXX	XXXX	0
LUI	01101	101	XXX	XXXX	TBD
JALR	11001	11X	XXX	XXXX	TBD

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

- Εντολές ολίσθησης:
  - *Τύπου Register (R)*
    - `SLL  Rd, Rs1, Rs2` `Rd = Rs1 <<  Rs2(4:0)`
    - `SRL  Rd, Rs1, Rs2` `Rd = Rs1 >>  Rs2(4:0)`
    - `SRA  Rd, Rs1, Rs2` `Rd = Rs1 >>> Rs2(4:0)`
  - *Τύπου Immediate (I)*
    - `SLLI Rd, Rs1, uimm` `Rd = Rs1 <<  imm(4:0)`
    - `SRLI Rd, Rs1, uimm` `Rd = Rs1 >>  imm(4:0)`
    - `SRAI Rd, Rs1, uimm` `Rd = Rs1 >>> 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 ScrB 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 (= shiftRorI)

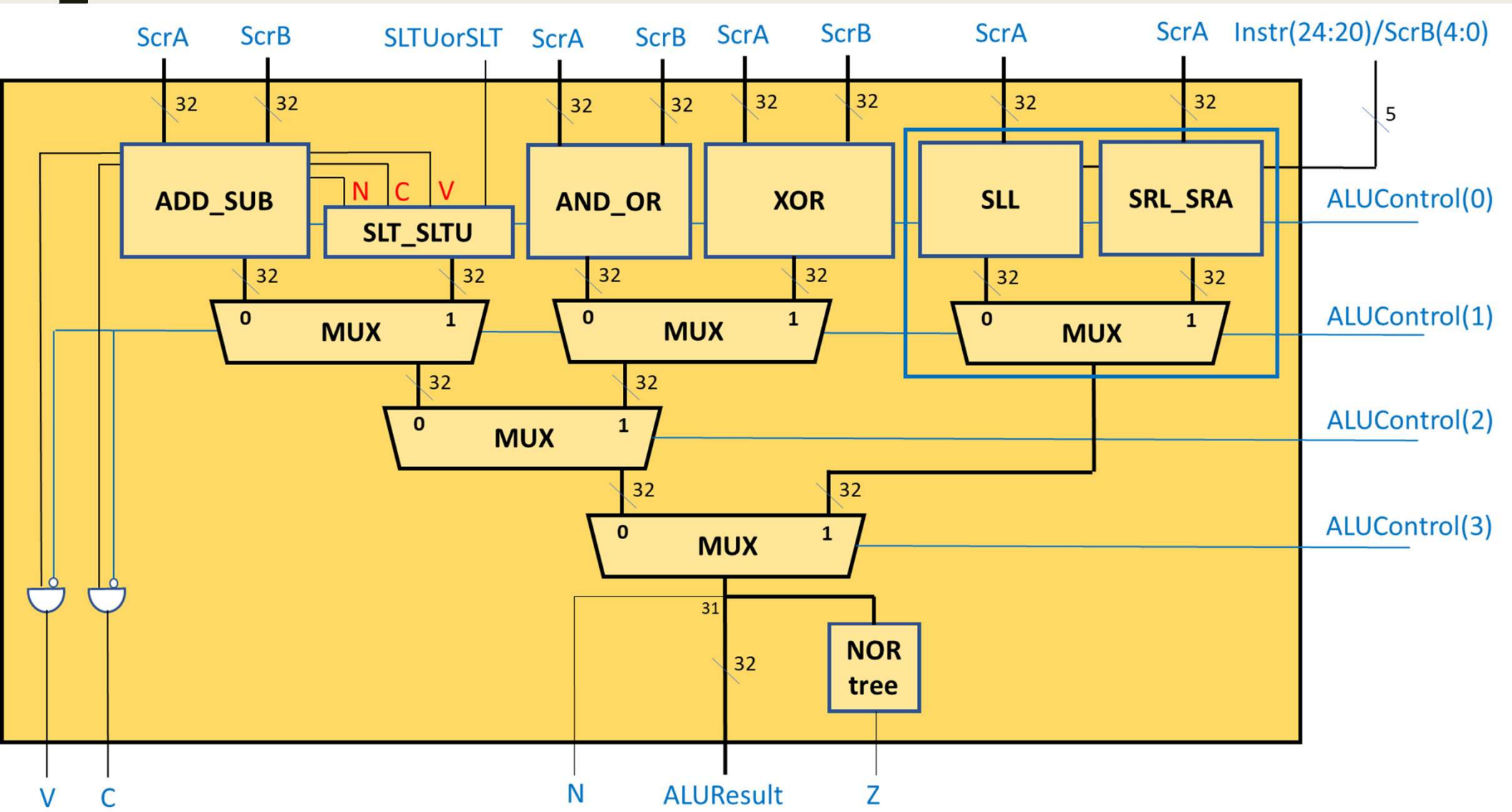
Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

- 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

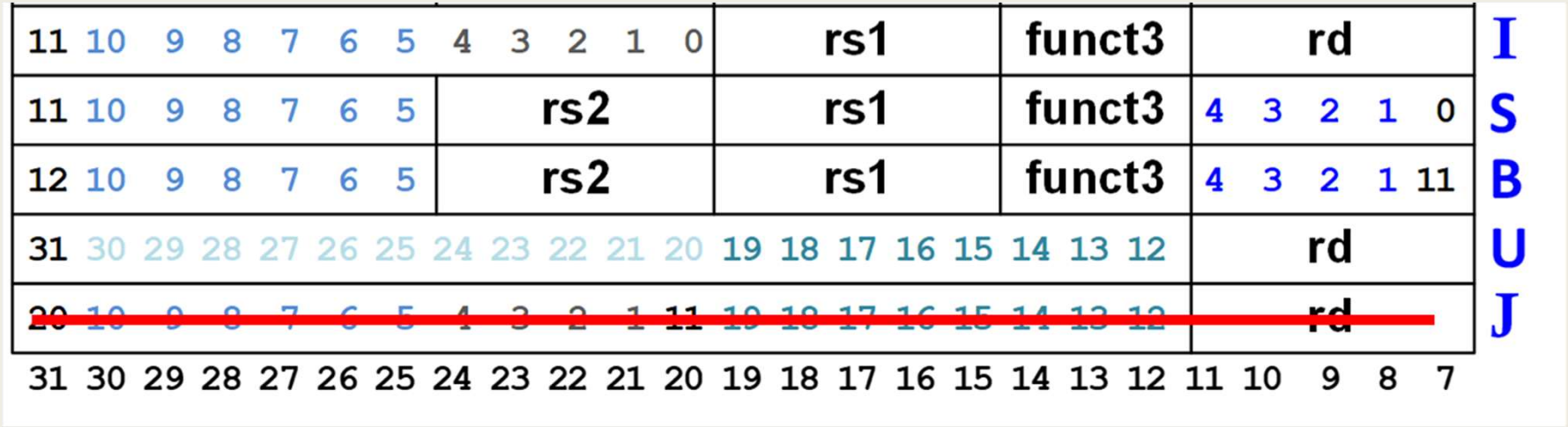
Τύπος	ImmSrc	Extend (31:0) =	Input
I	00	**`Sign_Ext(instr(31:20))`**	12 bit
S	01	**`Sign_Ext(instr(31:25)&instr(11:7))`**	12 bit
B	10	**`Sign_Ext(instr(31)&instr(7) &instr(30:25)&instr(11:8)&'0')`**	13 bit
U	11	**`instr(31:12)&0x000`**	20 bit

    - 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

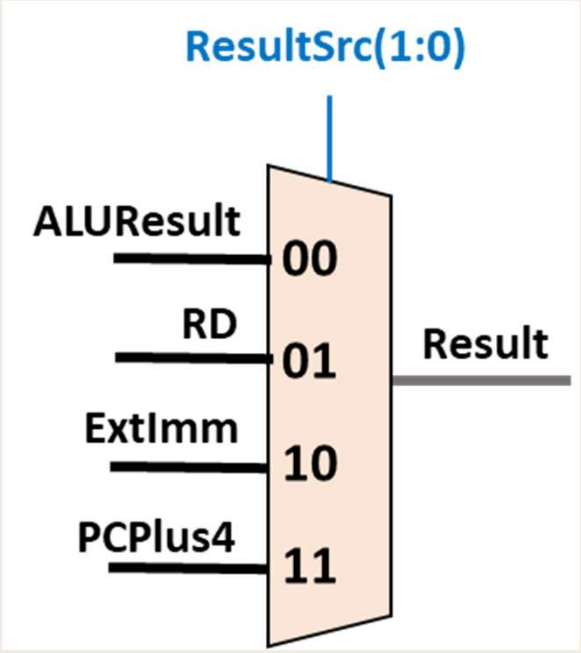
- Για το βασικό σύνολο εντολών της αρχιτεκτονικής RISC-V, που έχουμε μελετήσει μέχρι τώρα, η μονάδα Extend εκτελεί τις ακόλουθες διακριτές επεκτάσεις προσήμου στα 32 bit:

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



Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

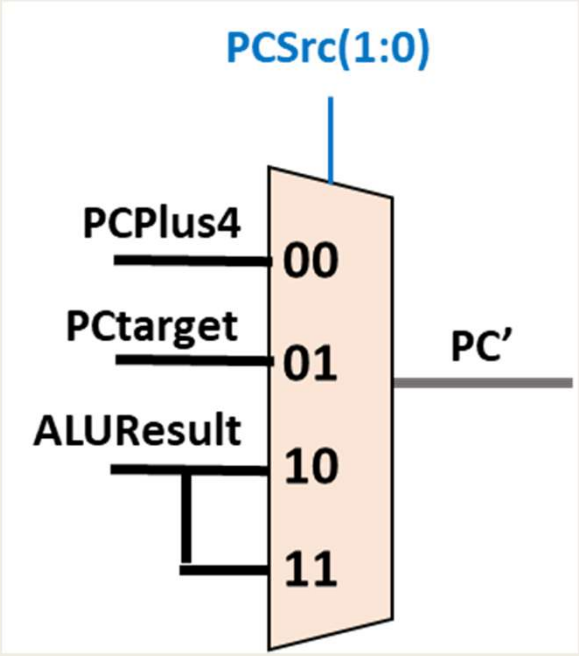
- Εντολή JALR/JR:
  - *Τύπου Immediate (I)*
    - **`JALR  Rd, Rs1, imm`** **`PC = Rs1 + S_Ext(imm)`**
      `Rd = PC + 4`
  - *Προσθήκες στην διαδρομή δεδομένων*
    - επέκταση του πολυπλέκτη Result σε 4 σε 1, ώστε να επιλέγει μεταξύ:
      - *00: ALUResult, 01: RD (από DM), 10 ExtImm και 11: PCPlus4 για εγγραφή στο αρχείο καταχωρητών (**Rd**)*
      - *η επιλογή γίνεται με είσοδο επιλογής ResultSrc(1:0)*




```
JR  zero, ra, 0        PC = ra + 0
                       zero = PC + 4
```

Επεξεργαστής ενός κύκλου: πρόσθετες εντολές

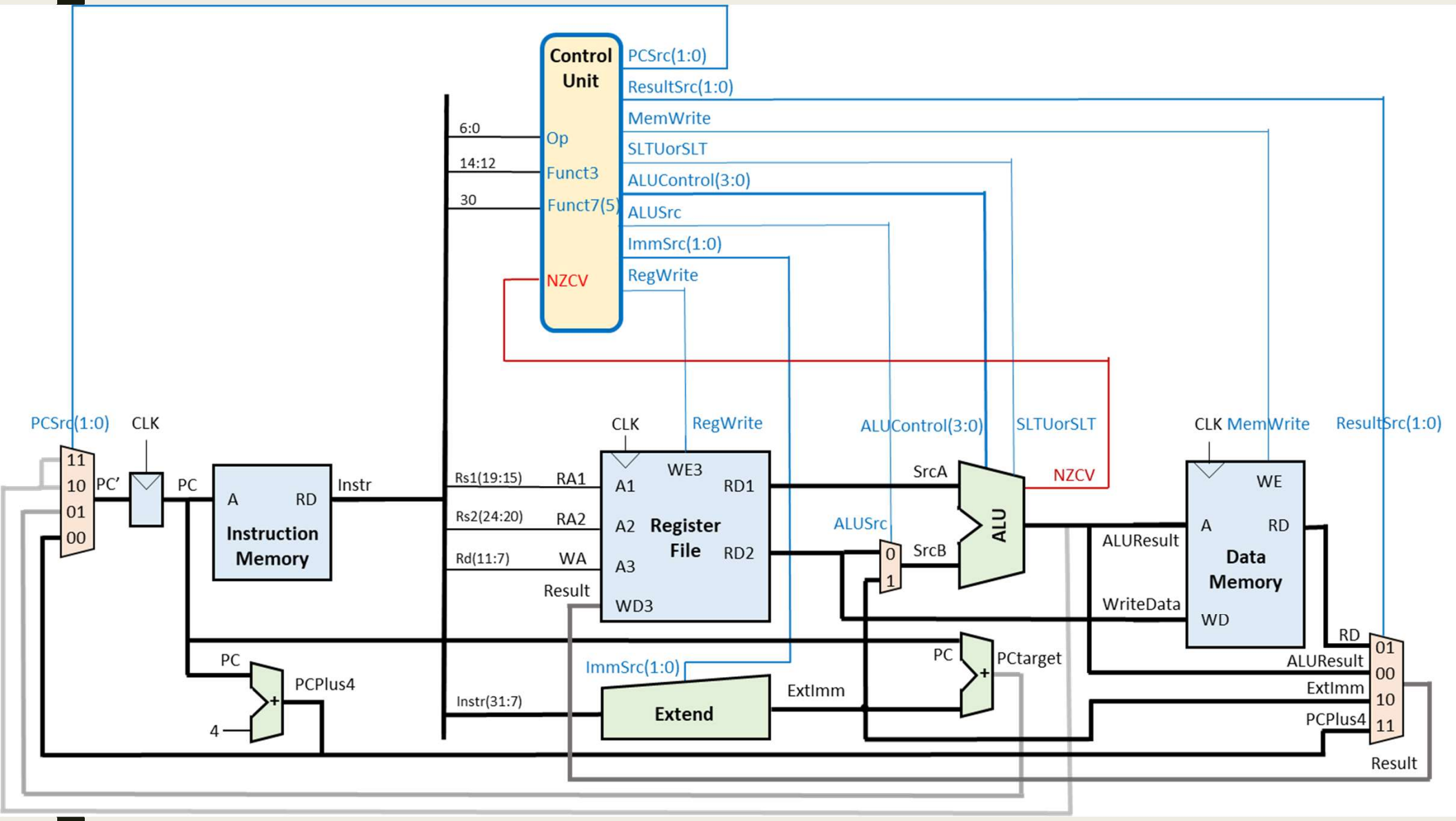
- Εντολή JALR/JR:
  - *Τύπου Immediate (I)*
    - **`JALR  Rd, Rs1, imm`** **`PC = Rs1 + S_Ext(imm)`**
      `Rd = PC + 4`
  - *Προσθήκες στην διαδρομή δεδομένων*
    - επέκταση του πολυπλέκτη PC’ σε 4 σε 1, ώστε να επιλέγει μεταξύ:
      - *00: PCPlus4, 01: PCtarget, 1X ALUResult*
        για εγγραφή στο μετρητή προγράμματος PC
      - *η επιλογή γίνεται με είσοδο επιλογής PCSrc(1:0)*




```
JR  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)

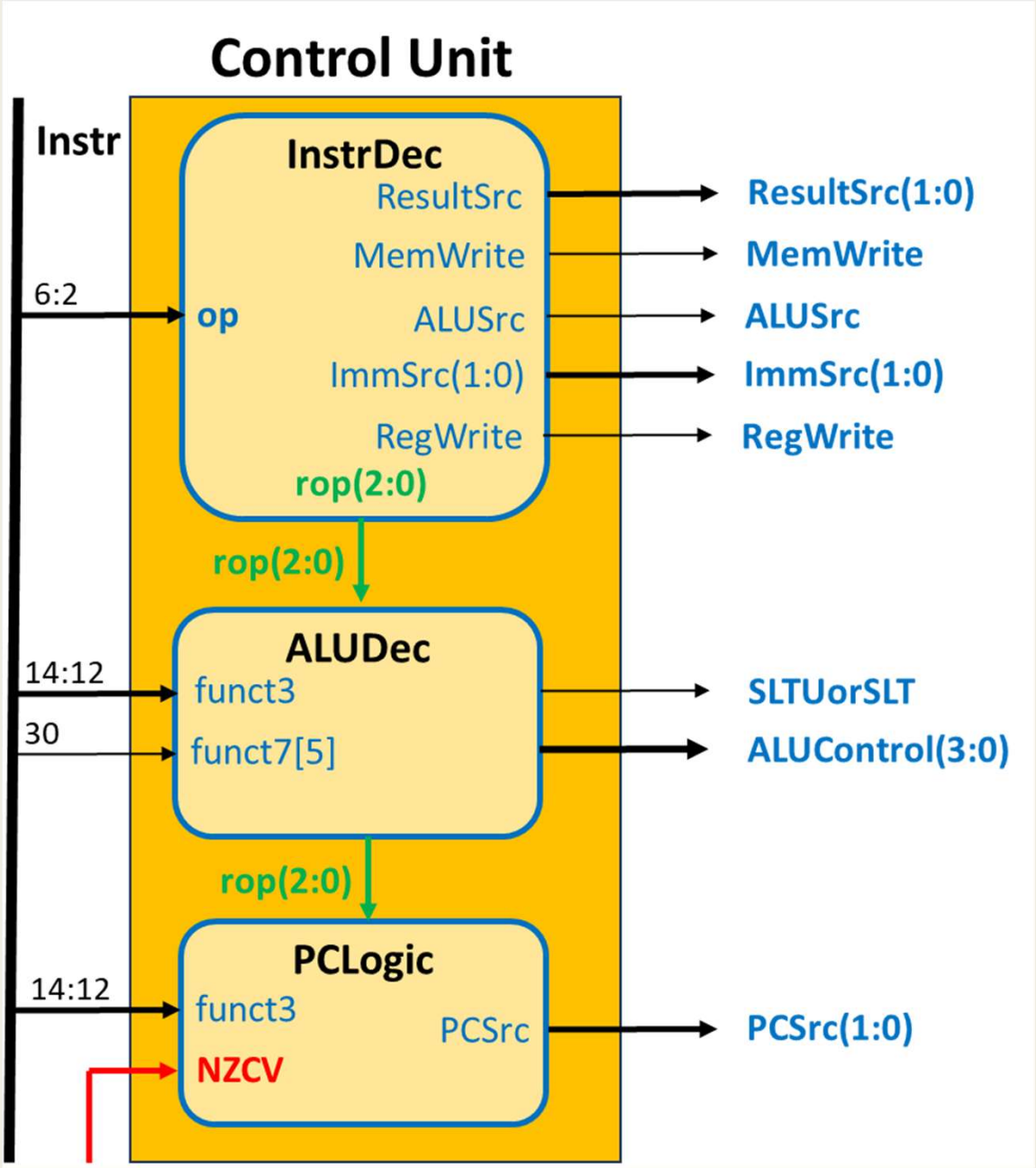


Επεξεργαστής ενός κύκλου: ολοκληρωμένη μονάδα ελέγχου RISC-V ενός κύκλου

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

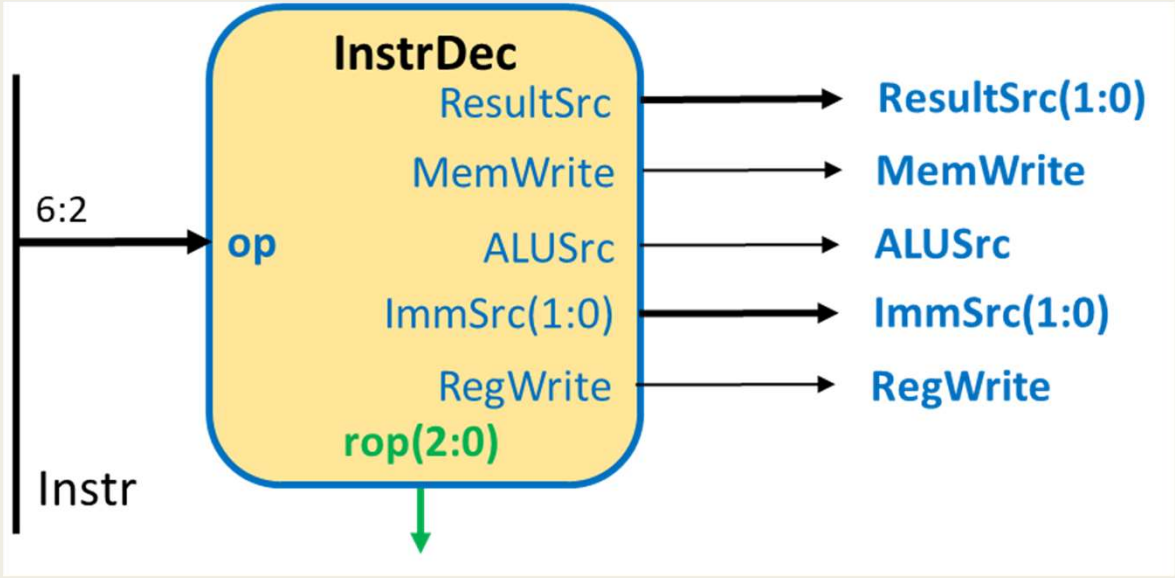
1. Κύριος αποκωδικοποιητής εντολής (InstrDec)
   - *η υπομονάδα που αποκωδικοποιεί το πεδίο op της εντολής (Instr(6:2) και παράγει τα σήματα επιλογής πολυπλεκτών ALUSrc και ResultSrc(1:0), το σήμα επιλογής επέκτασης προσήμου ImmSrc(1:0), τα σήματα ελέγχου εγγραφής στο αρχείο καταχωρητών RegWrite και 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 (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)=100, 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



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

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)*



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

1. Αποκωδικοποιητής εντολής (InstrDec)

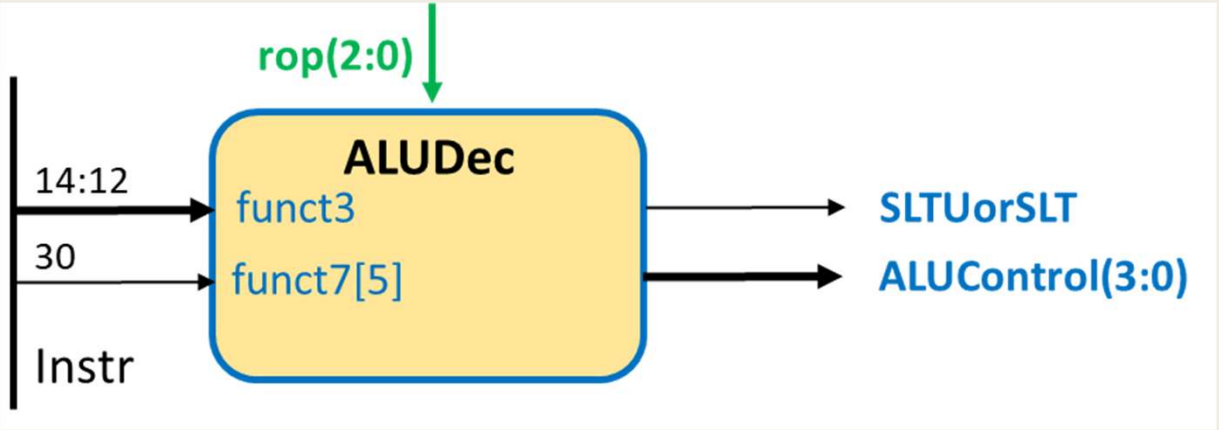
Εντολή	op(6:2)	rop(2:0)	Result Src(1:0)	Mem Write	ALU Src	Imm Src(1:0)	Reg Write
LW	00000	000	01	0	1	00	1
SW	01000	001	XX	1	1	01	0
ALU	01100	010	00	0	0	XX	1
ALUI	00100	011	00	0	1	00	1
Branch	11000	100	XX	0	0	10	0
LUI	01101	101	10	0	X	11	1
JALR	11001	11X	11	0	1	00	1

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(2:0) 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	0110011	010	000	0	add	0000	X
SUB	0110011	010	000	1	sub	0001	X
AND	0110011	010	111	0	and	0100	X
OR	0110011	010	110	0	or	0101	X
XOR	0110011	010	100	0	xol	011X	X
SLL	0110011	010	001	0	sll	1X0X	X
SRL	0110011	010	101	0	srl	1X10	X
SRA	0110011	010	101	1	sra	1X11	X
SLT	0110011	010	010	0	slt	001X	0
SLTU	0110011	010	011	0	sltu	001X	1

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

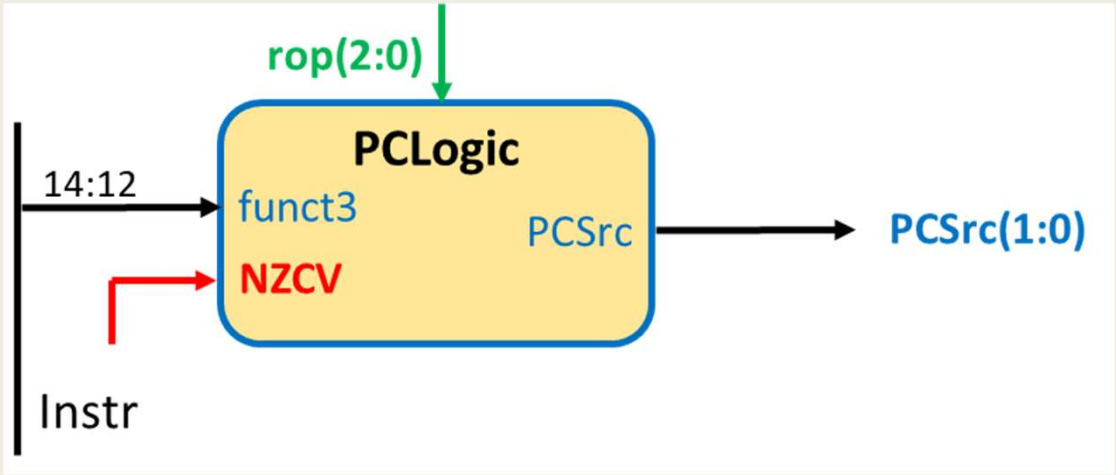
2. Αποκωδικοποιητής μονάδας ALU (ALUDec)

Εντολή	op(6:0)	Rop(2:0)	funct3	Funct7 (5)	ALU πράξη	ALUControl (3:0)	SLTUorSLT
ADDI	0010011	011	000	X	add	0000	X
ANDI	0010011	011	111	X	and	0100	X
ORI	0010011	011	110	X	or	0101	X
XORI	0010011	011	100	X	xol	011X	X
LW	0000011	000	010	X	add	0000	X
SW	0100011	001	010	X	add	0000	X
BEQ	1100011	100	000	X	sub	0001	X
BNE	1100011	100	001	X	sub	0001	X
LUI	0110111	101	XXX	X	-	XXXX	X
JALR	1100111	11X	000	X	add	0000	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)=100, 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(6:0)	rop(2:0)	funct3	NZCV	PCSrc
BEQ	1100011	100	000	X0XX	00
BEQ	1100011	100	000	X1XX	01
BNE	1100011	100	001	X0XX	01
BNE	1100011	100	001	X1XX	00
LW	00000	000	XXX	XXXX	00
SW	01000	001	XXX	XXXX	00
ALU	01100	010	XXX	XXXX	00
ALUI	00100	011	XXX	XXXX	00
LUI	01101	101	XXX	XXXX	00
JALR	11001	11X	XXX	XXXX	1X

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} = (# instructions/program) × CPI × T_c = 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
  - CPU_{time} = (# instructions/program) × CPI × T_c = 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

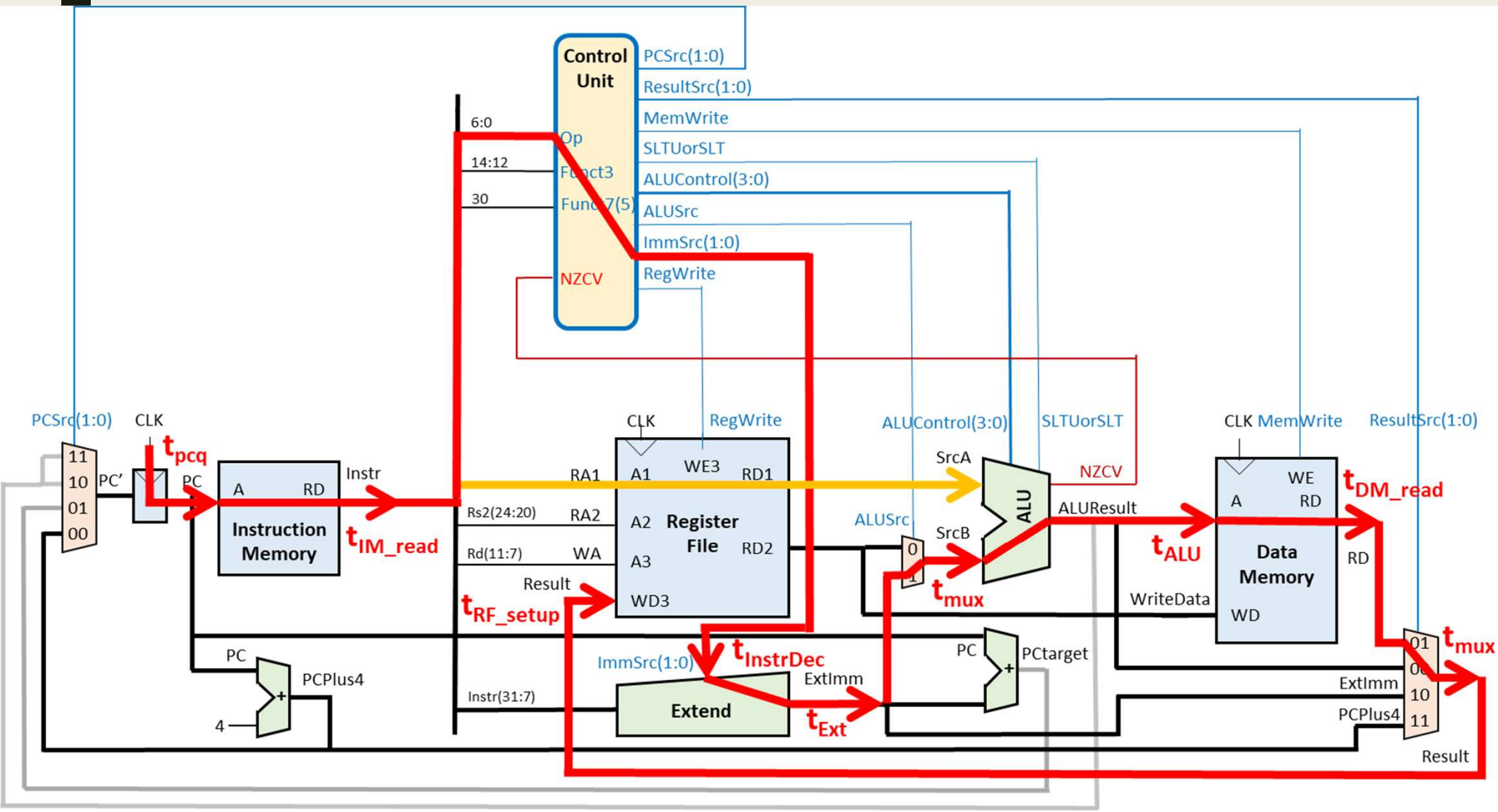
- Πώς επηρεάζεται η επίδοση
  - CPU_{time} = (# instructions/program) × CPI × T_c = seconds/program
- Μέσος αριθμός κύκλων ανά εντολή (cycles per instruction - CPI):
  - *μέσος αριθμός κύκλων ανά εντολή του ρολογιού που απαιτούνται για την εκτέλεση μιας εντολής ενός προγράμματος*
  - *εξαρτάται από το 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_{mux})
  - 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 ως Result μέσω πολυπλέκτη 4 σε 1 (t_{mux})
  - 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
  - $T_c = t_{pcq} + t_{IM\_read} + \max(t_{RF\_read}, (t_{InstrDec} + t_{Ext} + t_{mux2})) + t_{ALU} + t_{DM\_read} + t_{mux4} + t_{RF\_setup}$



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
  - $T_c = t_{pcq} + t_{IM\_read} + \max(t_{RF\_read}, (t_{InstrDec} + t_{Ext} + t_{mux2})) + t_{ALU} + t_{DM\_read} + t_{mux4} + t_{RF\_setup}$
- Παράδειγμα υπολογισμού επιδόσεων επεξεργαστή ενός κύκλου:
  - *η περίοδος του ρολογιού (CLK) με ενδεικτικές καθυστερήσεις είναι:*
    - $T_c = (40 + 200 + (60 + 60 + 40) + 250 + 300 + 60 + 60)$ ps = 1070 ps
  - ο χρόνος εκτέλεσης ενός προγράμματος των 100 δις εντολών είναι:
    - $CPU_{time} = (\#\ instr/program) \times CPI \times T_c = 100 \times 10^9 \times 1 \times 1070 \times 10^{-12} =$ 107 s

Ενδεικτικές τιμές. Υποθέτουμε $t_{RF\_read} < t_{InstrDec} + t_{Ext} + t_{mux}$

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
Καθυστέρηση διάδοσης στον αποκωδικοποιητή εντολών	$t_{InstrDec}$	60
Καθυστέρηση διάδοσης (control path) στη μονάδα Extend	t_{Ext}	60
Καθυστέρηση διάδοσης πολυπλέκτη 2σε1 και 4σε1	T_{mux2}/T_{mux4}	40/60
Καθυστέρηση διάδοσης στη μονάδα ALU για πρόσθεση	t_{ALU}	250
Χρόνος ανάγνωσης από τη μνήμη εντολών (ROM)	t_{IM_read}	200
Χρόνος ανάγνωσης από τη μνήμη δεδομένων (Distr.-RAM)	t_{DM_read}	250
Χρόνος ανάγνωσης από το αρχείο καταχωρητών (Distr.-RAM)	t_{RF_read}	150