ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Σχεδίαση Ψηφιακών Συστημάτων

Ενότητα 3

Σχεδίαση της Διόδου Δεδομένων

Καθηγητής Αντώνης Πασχάλης

2010

Γενικές Γραμμές

- Επεξεργαστής (Δίοδος Δεδομένων και Μονάδα Ελέγχου)
- Μνήμη Εντολών και Μνήμη Δεδομένων (Cache L1)
- Καταχωρητές Ειδικής Χρήσης της Διόδου Δεδομένων
- Διαδικασία Σχεδίασης Διόδου Δεδομένων
- Αφαιρετική Ανάλυση των Εντολών του MIPS R2000
- Φάσεις Εκτέλεσης Εντολών
- Υλοποίηση των Λειτουργικών Μονάδων της Διόδου Δεδομένων
- Λεπτομερής Ανάλυση των Εντολών του MIPS R2000 σε Μικρο-Λειτουργίες
- Προσδιορισμός των Σημάτων Ελέγχου
- Δίοδος Δεδομένων 5 Κύκλων του MIPS R2000
- Εκτέλεση των Εντολών με Βάση τη Δίοδο Δεδομένων

Επεξεργαστής - CPU

- **Δίοδος Δεδομένων (Datapath)**
  - χρησιμοποιείται για την εκτέλεση των εντολών, η οποία ανάγεται στην εκτέλεση μίας ακολουθίας στοιχειωδών λειτουργιών που στη συνέχεια ονομάζουμε **μικρο-λειτουργίες**
- **Μονάδα Ελέγχου**
  - παράγει τα κατάλληλα **σήματα ελέγχου** που απαιτούνται για το συγχρονισμό όλων των μονάδων του υπολογιστή
  - αποκωδικοποιεί την εντολή που θα εκτελεσθεί και παράγει για κάθε **μικρο-λειτουργία** τα αντίστοιχα **σήματα ελέγχου** την κατάλληλη χρονική στιγμή

Δίοδος Δεδομένων (Datapath)

- **Λειτουργικές μονάδες**
  - **Αριθμητική & Λογική Μονάδα (ALU)**
    - Οι αριθμητικές και λογικές πράξεις που υλοποιεί εξαρτώνται από το σύνολο των αριθμητικών και λογικών εντολών που υποστηρίζει ο επεξεργαστής
    - Επίσης εξετάζει εάν είναι μηδέν το αποτέλεσμα μίας πράξης
  - **Επιπλέον Αθροιστές**
    - Για παράδειγμα υπολογίζει τη νέα τιμή του PC : PC <= PC + 4
  - **Κυκλώματα επέκτασης (μηδενός & πρόσημου)**
    - Για τον υπολογισμό του sign_extend(immediate) και του zero_extend(immediate)
- **Αρχείο καταχωρητών γενικής χρήσης (register file)**
  - **32 καταχωρητές των 32 ψηφίων**

Δίοδος Δεδομένων (Datapath)

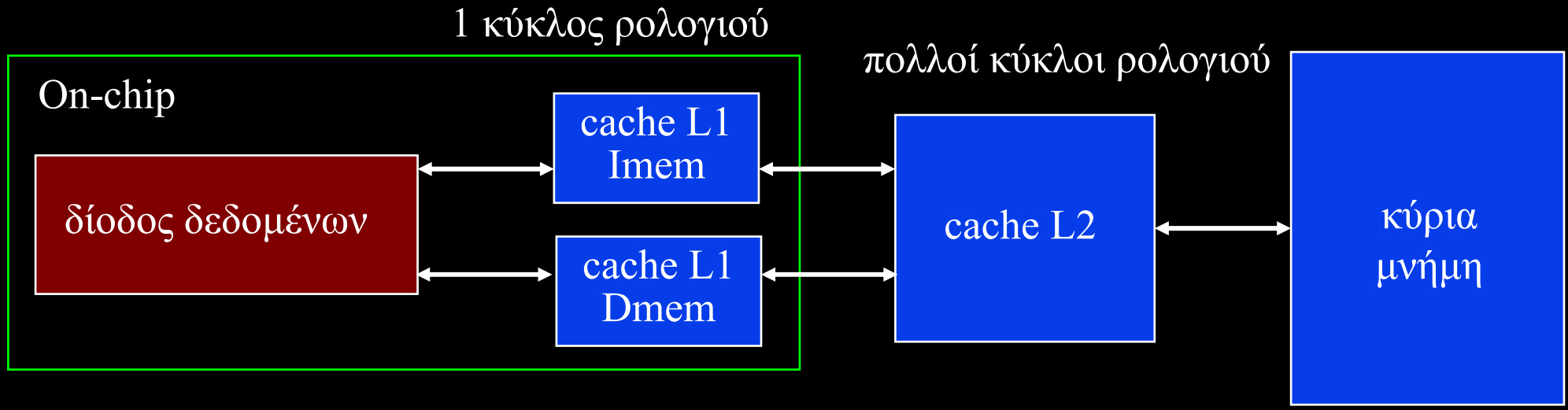
- **Εσωτερικές αρτηρίες** (internal buses)
  - για τη διασύνδεση των λειτουργικών μονάδων με τους καταχωρητές γενικής και ειδικής χρήσης
- **Πολυπλέκτες** (muxes)
  - για τον έλεγχο από τη μονάδα ελέγχου της ροής των πληροφοριών στις εσωτερικές αρτηρίες
- **Εξωτερικές Αρτηρίες**
  - για τη διασύνδεση της διόδου δεδομένων με το πρώτο επίπεδο μνήμης (cache L1)
  - για κάθε μνήμη απαιτείται μία αρτηρία διευθύνσεων και μία ή δύο αρτηρίες δεδομένων (για εγγραφή και διάβασμα) (address bus & data bus)

Μονάδα Ελέγχου

- Στις διόδους δεδομένων του ενός κύκλου υλοποιείται σαν συνδυαστικό κύκλωμα
- Στις διόδους δεδομένων των πολλών κύκλων υλοποιείται σαν σύγχρονη ακολουθιακή μηχανή με δύο τεχνικές:
  - καλωδιωμένη (hardwired) – για RISC
    σαν μηχανή πεπερασμένων καταστάσεων (FSM)
  - μικρο-προγραμματιζόμενη (microprogrammed)
    με χρήση μίας μνήμης ROM, όπου αποθηκεύονται μικρο-εντολές (microinstructions), και ενός μετρητή μικρο-προγράμματος (microprogram sequencer) – για CISC

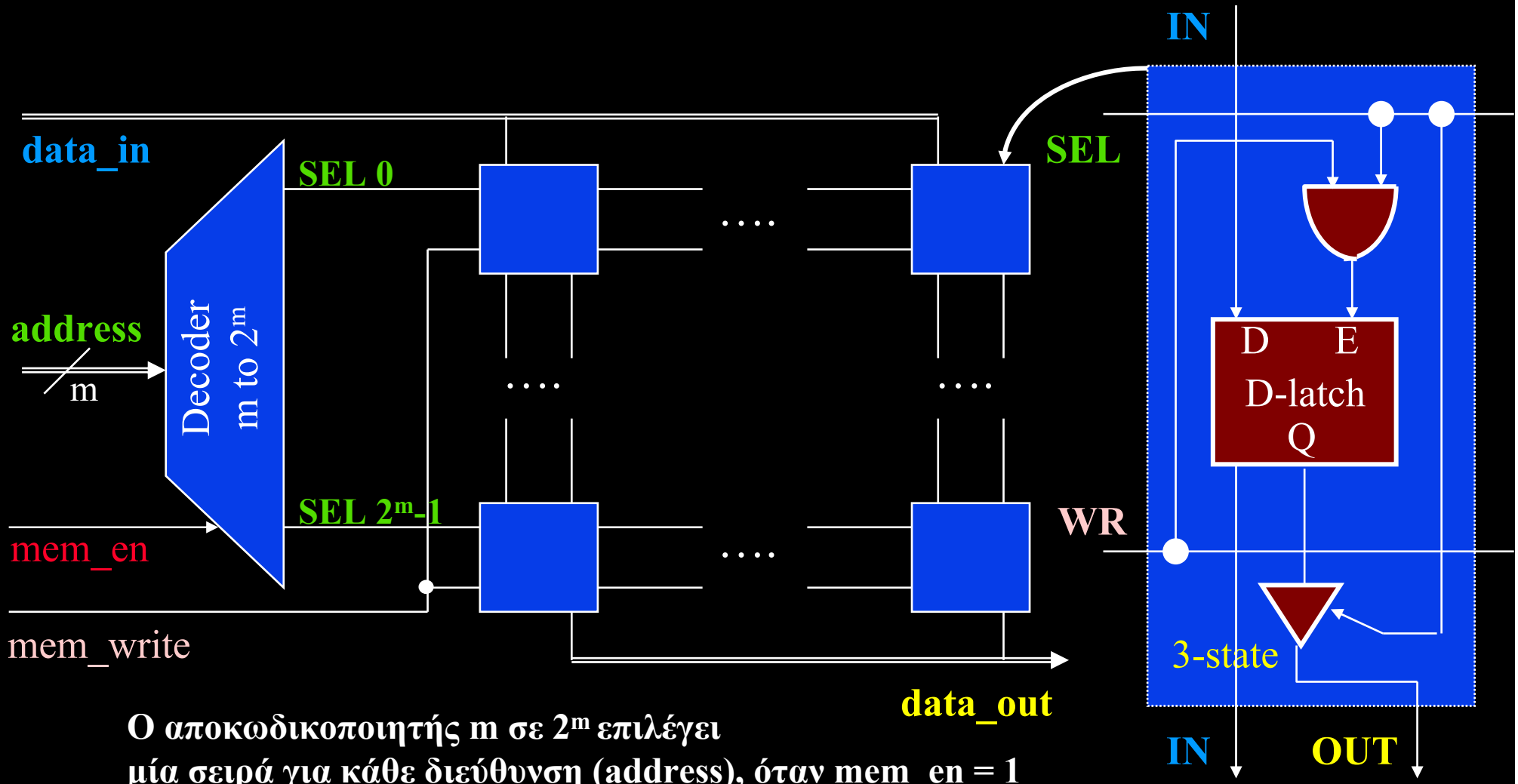
Υλοποίηση Μνήμης Cache L1

- **Αρχιτεκτονική Harvard στο Επίπεδο 1**
  Δύο χωριστές μνήμες Fast Static Random Access Memories (FSRAM)
  - **Μνήμη Εντολών Imem**
  - **Μνήμη Δεδομένων Dmem**



Ιεραρχία Συστήματος Μνήμης με 2 Επίπεδα Cache

Στατικές Μνήμες RAM - Cache L1



Ο αποκωδικοποιητής m σε 2^m επιλέγει
μία σειρά για κάθε διεύθυνση (address), όταν mem_en = 1
mem_write = 0, mem_en = 1 : διάβασμα από τη μνήμη
mem_write = 1-σε-0, mem_en = 1 : εγγραφή στη μνήμη

Δίοδος Δεδομένων (Datapath)

Καταχωρητές Ειδικής Χρήσης

- **μετρητής προγράμματος (program counter - PC)**
  - αποθηκεύει τη διεύθυνση της εντολής που πρόκειται να εκτελεσθεί και διευθυνσιοδοτεί τη μνήμη εντολών
- **καταχωρητής εντολών (instruction register - IR)**
  - αποθηκεύει την εντολή που πρόκειται να εκτελεσθεί, που διαβάζεται από τη μνήμη εντολών

Υποθέτουμε ότι υπάρχει χωριστή μνήμη εντολών (Imem) και χωριστή μνήμη δεδομένων (Dmem)
Αρχιτεκτονική Harvard

Δίοδος Δεδομένων (Datapath)

Καταχωρητές Ειδικής Χρήσης

- **καταχωρητής διευθύνσεων μνήμης (memory address register - MAR)**
  - αποθηκεύει τη διεύθυνση του τελεστέου που πρέπει να διαβασθεί από τη μνήμη δεδομένων ή του αποτελέσματος που πρέπει να αποθηκευθεί στη μνήμη δεδομένων
- **καταχωρητής δεδομένων μνήμης (memory data register - MDR)**
  - αποθηκεύει τον τελεστέο που διαβάζεται από τη μνήμη δεδομένων ή το αποτέλεσμα που γράφεται στη μνήμη δεδομένων

Υποθέτουμε ότι υπάρχει χωριστή μνήμη δεδομένων (Dmem) με μία αμφίδρομη αρτηρία δεδομένων

Δίοδος Δεδομένων (Datapath)

Καταχωρητές Ειδικής Χρήσης

- **καταχωρητής δεδομένων μνήμης που διαβάζονται από τη μνήμη δεδομένων (memory data-out register - MDR out)**
  - αποθηκεύει τον τελεστέο που διαβάζεται από τη μνήμη δεδομένων
- **καταχωρητής δεδομένων μνήμης που γράφονται στη μνήμη δεδομένων (memory data-in register - MDR in)**
  - αποθηκεύει το αποτέλεσμα που γράφεται στη μνήμη δεδομένων

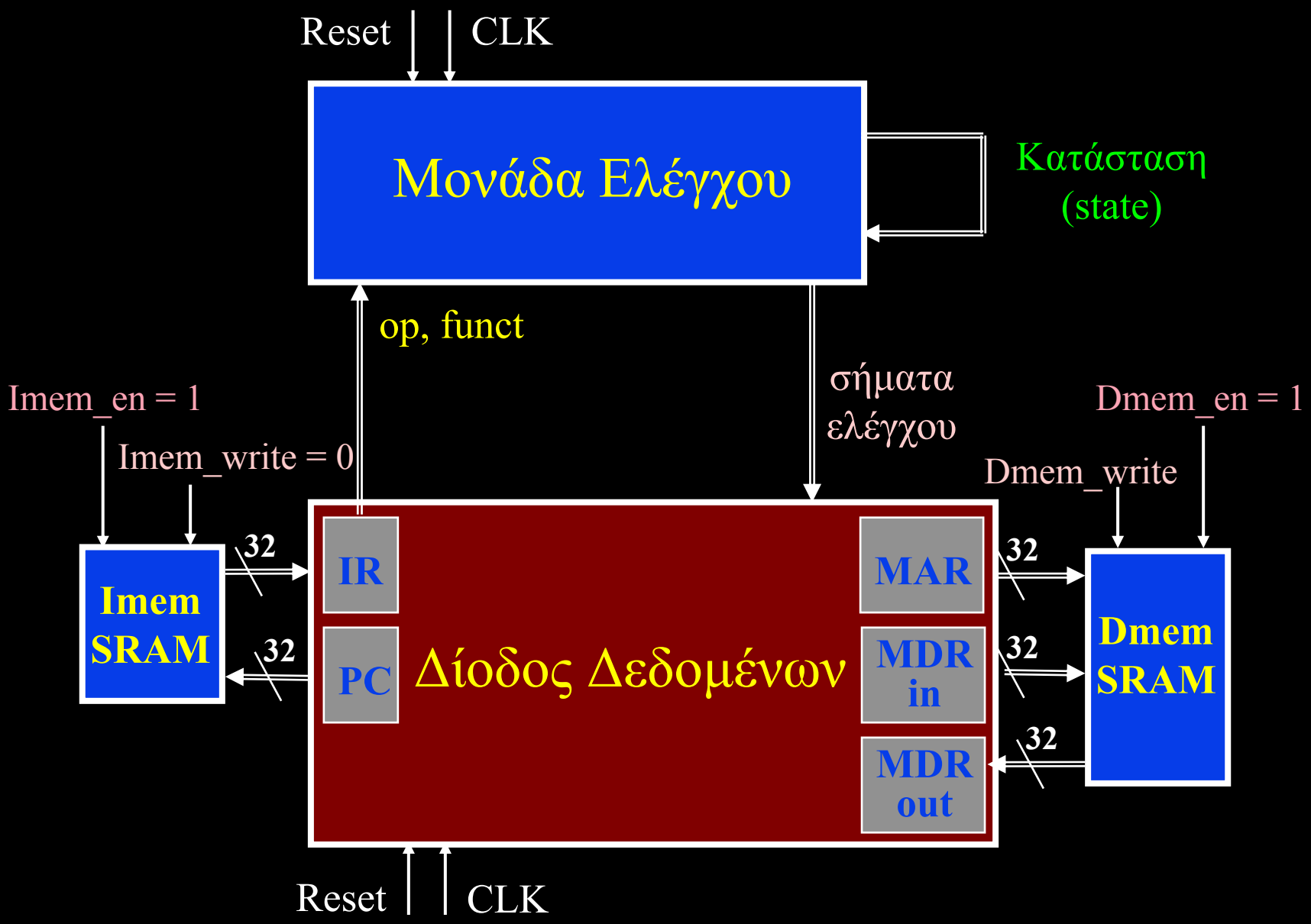
Υποθέτουμε ότι υπάρχει χωριστή μνήμη δεδομένων (Dmem) με δύο αρτηρίες δεδομένων

Δίοδος Δεδομένων (Datapath)

Καταχωρητές Ειδικής Χρήσης

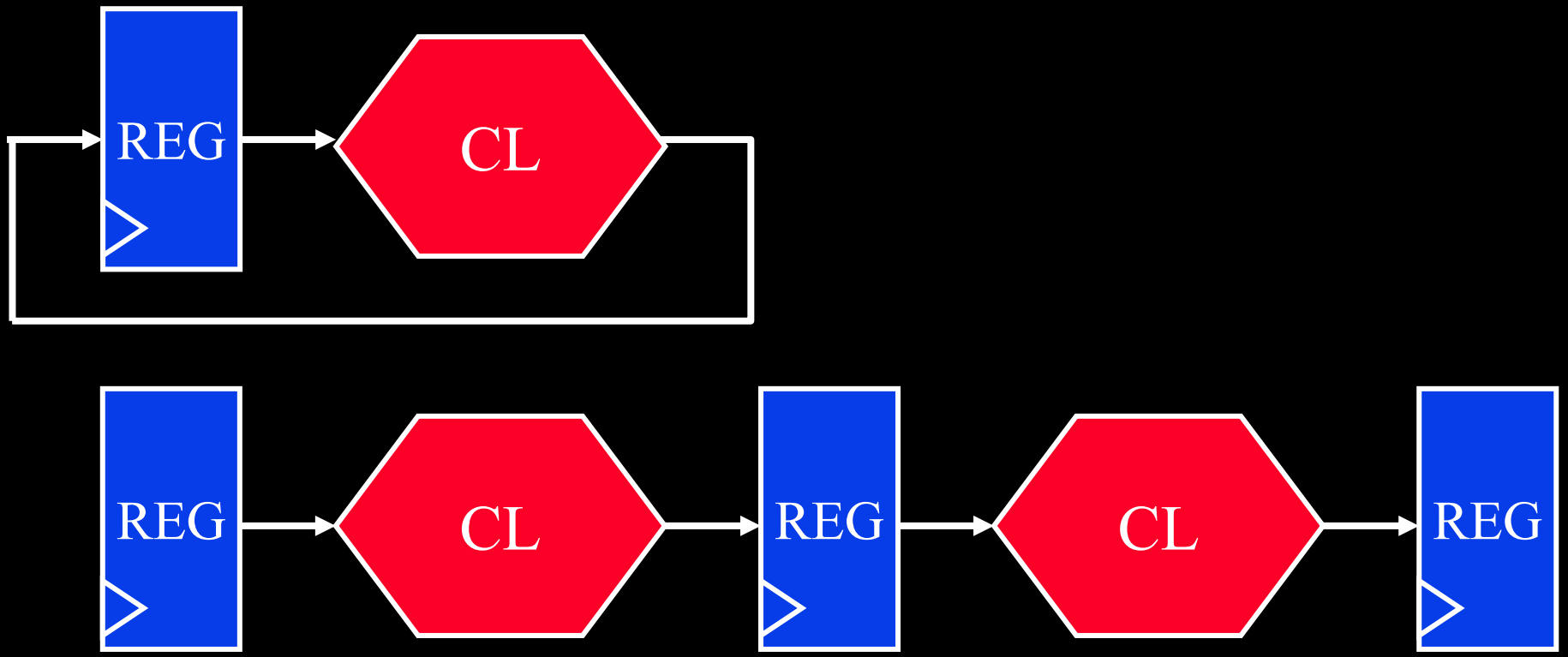
- **προσωρινοί καταχωρητές**
  - αποθηκεύουν προσωρινά την πληροφορία, που παράγεται κατά την εκτέλεση των μικρο-λειτουργιών, ώστε να είναι δυνατή η εκτέλεση μίας εντολής σε πολλούς κύκλους ρολογιού
  - απαιτείται συνήθως ένας κύκλος ρολογιού για κάθε μικρο-λειτουργία που δεν μπορεί να εκτελεσθεί παράλληλα με μία άλλη μικρο-λειτουργία
  - είναι προαιρετικοί

Επεξεργαστής - CPU



Περιγραφή Επεξεργαστή σε Επίπεδο Μεταφοράς Καταχωρητή - RTL

- περιγράφεται κάθε **καταχωρητής (REG)** του συστήματος, καθώς και η **συνδυαστική λογική (CL)** ανάμεσα στους καταχωρητές



Πως Σχεδιάζουμε Διόδους Δεδομένων ;

- Αναλύουμε το σύνολο των εντολών και προσδιορίζουμε τις **απαιτήσεις σε υλικό** της διόδου δεδομένων
  - από περιγραφές των εντολών σε επίπεδο μεταφοράς καταχωρητών (register transfer level - RTL) προσδιορίζουμε:
    - τις απαιτήσεις σε **καταχωρητές**
    - τις απαιτήσεις σε **λειτουργικές μονάδες**
    - τις **μικρο-λειτουργίες**
- Επιλέγουμε σε **πόσους κύκλους** θα εκτελεσθεί η εντολή και προσδιορίζουμε τους **προσωρινούς καταχωρητές**
- **Συνθέτουμε τη δίοδο των δεδομένων**

Αφαιρετική Ανάλυση των Εντολών του MIPS R2000

- Γίνεται σε **υψηλό** επίπεδο μεταφοράς καταχωρητή
- Χρησιμοποιούνται ως καταχωρητές
  - οι καταχωρητές ειδικού σκοπού της διόδου δεδομένων που ήδη έχουμε ορίσει (**PC, IR, MAR, MDR_in, MDR_out**)
  - οι καταχωρητές **reg[rs/rt/rd]** του αρχείου των 32 καταχωρητών, που προσδιορίζονται στα αντίστοιχα πεδία διευθύνσεων καταχωρητών (rs, rt, rd) του IR
  - το πεδίο **immediate** του IR στις εντολές τύπου I
  - το πεδίο **shamt** του IR στις εντολές ολίσθησης με σταθερό αριθμό ολισθήσεων
  - οι **θέσεις μνήμης της μνήμης εντολών** μεγέθους 4 bytes των οποίων η διεύθυνση του περισσότερου σημαντικού byte προσδιορίζεται στο **PC**
  - οι **θέσεις μνήμης της μνήμης δεδομένων** μεγέθους 4 bytes των οποίων η διεύθυνση του περισσότερου σημαντικού byte προσδιορίζεται στο **MAR**

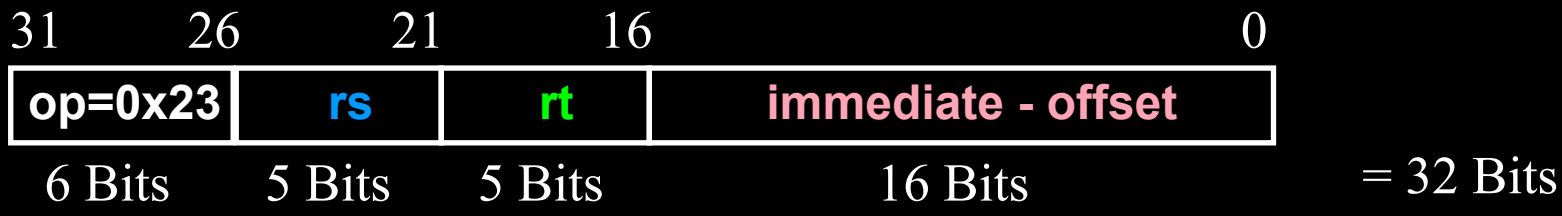
Σημειογραφία

- Το περιεχόμενο του καταχωρητή ειδικής χρήσης X αναφέρεται σαν **X**
- Το περιεχόμενο του καταχωρητή reg[rx] του αρχείου των 32 καταχωρητών αναφέρεται σαν **reg[rx]**
- Το περιεχόμενο του πεδίου immediate του IR αναφέρεται σαν **immediate**
  - μετά την επέκταση πρόσημου αναφέρεται σαν **sign_extend(immediate)**
  - μετά την επέκταση μηδενός αναφέρεται σαν **zero_extend(immediate)**
- Το περιεχόμενο του πεδίου shamt του IR αναφέρεται σαν **shamt**
- Το περιεχόμενο της θέσης μνήμης εντολών Imem στη διεύθυνση PC αναφέρεται σαν **Imem[PC]**
- Το περιεχόμενο της θέσης μνήμης δεδομένων Dmem στη διεύθυνση MAR αναφέρεται σαν **Dmem[MAR]**

Εντολή LW (Load Word)

- LW rt, immediate(rs) (= address)

```
IR <= Imem[PC]
MAR <= sign_extend(immediate) + reg[rs]
MDR_out <= Dmem[MAR]
reg[rt] <= MDR_out
PC <= PC + 4
```

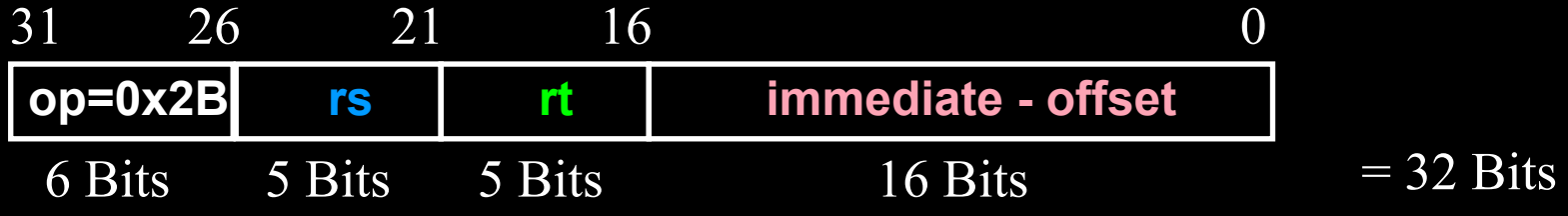


Υποθέτουμε ότι υπάρχει χωριστή μνήμη δεδομένων (Dmem) με δύο αρτηρίες δεδομένων και με MAR, MDR_in και MDR_out

Εντολή SW (Store Word)

- SW rt, immediate(rs) (= address)

```
IR <= Imem[PC]
MAR <= sign_extend(immediate) + reg[rs]
MDR_in <= reg[rt]
DMem[MAR] <= MDR_in
PC <= PC + 4
```



Υποθέτουμε ότι υπάρχει χωριστή μνήμη δεδομένων (Dmem) με δύο αρτηρίες δεδομένων και με MAR, MDR_in και MDR_out

Εντολή ADDIU (Addition Immediate Unsigned)

- **ADDIU rt, rs, immediate**

```
IR <= Imem[PC]
reg[rt] <= reg[rs] + sign_extend(immediate)
PC <= PC + 4
```

Η πρόσθεση γίνεται μεταξύ προσημασμένων ακεραίων αριθμών, χωρίς να γίνεται έλεγχος υπερχείλισης

Εντολή ANDI (AND Immediate)

- ANDI rt, rs, immediate

```
IR <= Imem[PC]
reg[rt] <= reg[rs] AND zero_extend(immediate)
PC <= PC + 4
```

Εντολή ORI (OR Immediate)

- ORI rt, rs, immediate

```
IR <= Imem[PC]
reg[rt] <= reg[rs] OR zero_extend(immediate)
PC <= PC + 4
```

Εντολή XORI (XOR Immediate)

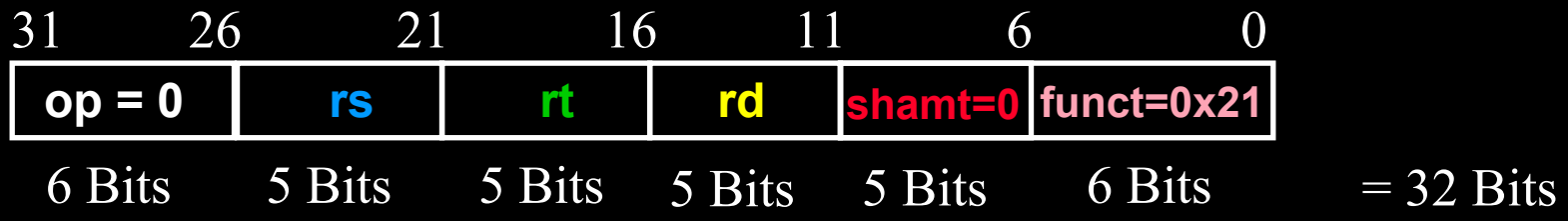
- **XORI rt, rs, immediate**

```
IR <= Imem[PC]
reg[rt] <= reg[rs] XOR zero_extend(immediate)
PC <= PC + 4
```

Εντολή ADDU (Addition Unsigned)

- ADDU rd, rs, rt

```
IR <= Imem[PC]
reg[rd] <= reg[rs] + reg[rt]
PC <= PC + 4
```

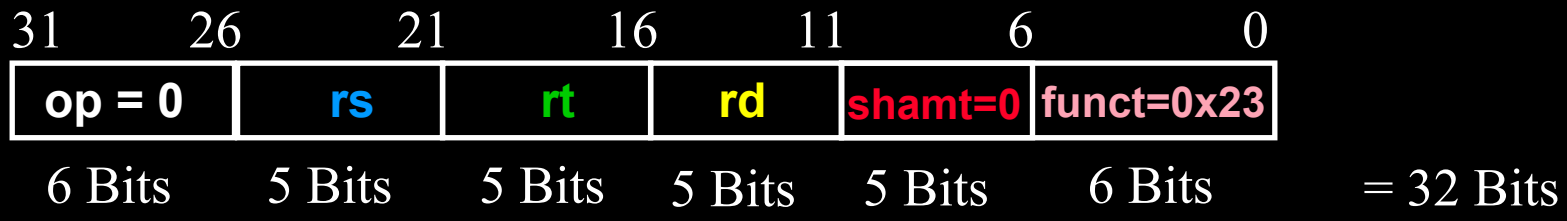


Η πρόσθεση γίνεται μεταξύ προσημασμένων ακεραίων αριθμών, χωρίς να γίνεται έλεγχος υπερχείλισης

Εντολή SUBU (Subtraction Unsigned)

- SUBU rd, rs, rt

```
IR <= Imem[PC]
reg[rd] <= reg[rs] - reg[rt]
PC <= PC + 4
```

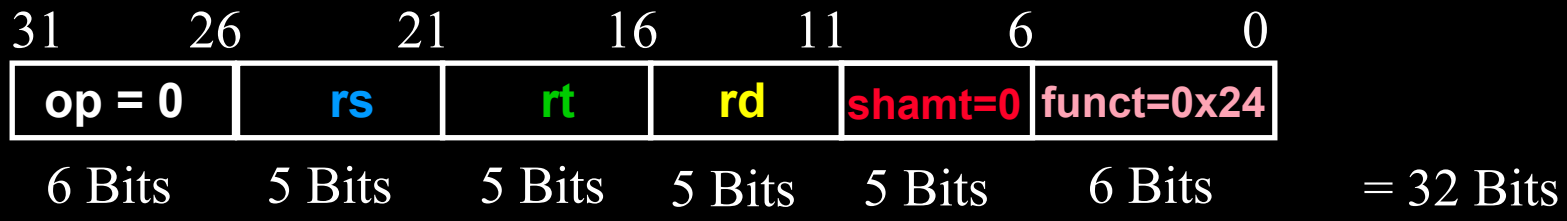


Η αφαίρεση γίνεται μεταξύ προσημασμένων ακεραίων αριθμών, χωρίς να γίνεται έλεγχος υπερχείλισης

Εντολή AND

- AND rd, rs, rt

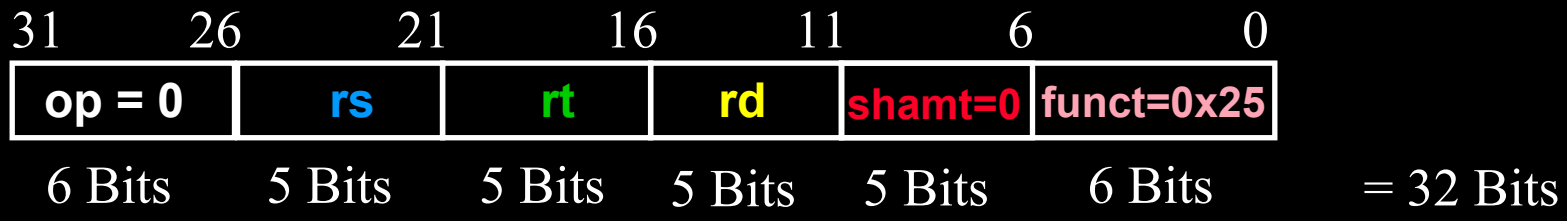
```
IR <= Imem[PC]
reg[rd] <= reg[rs] AND reg[rt]
PC <= PC + 4
```



Εντολή OR

- OR rd, rs, rt

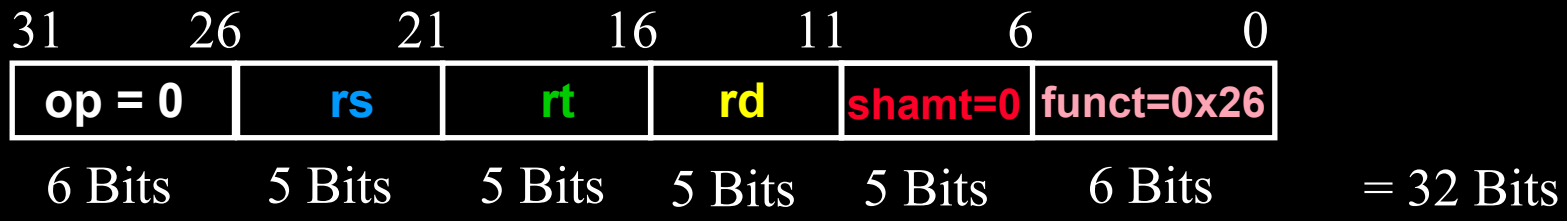
```
IR <= Imem[PC]
reg[rd] <= reg[rs] OR reg[rt]
PC <= PC + 4
```



Εντολή XOR

- XOR rd, rs, rt

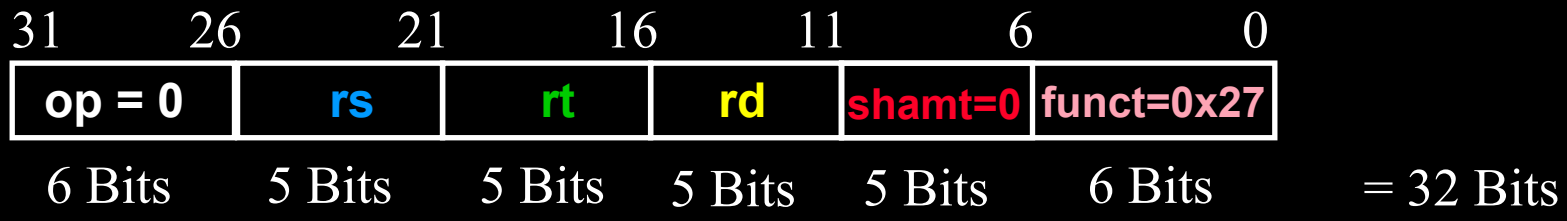
```
IR <= Imem[PC]
reg[rd] <= reg[rs] XOR reg[rt]
PC <= PC + 4
```



Εντολή NOR

- NOR rd, rs, rt

```
IR <= Imem[PC]
reg[rd] <= reg[rs] NOR reg[rt]
PC <= PC + 4
```



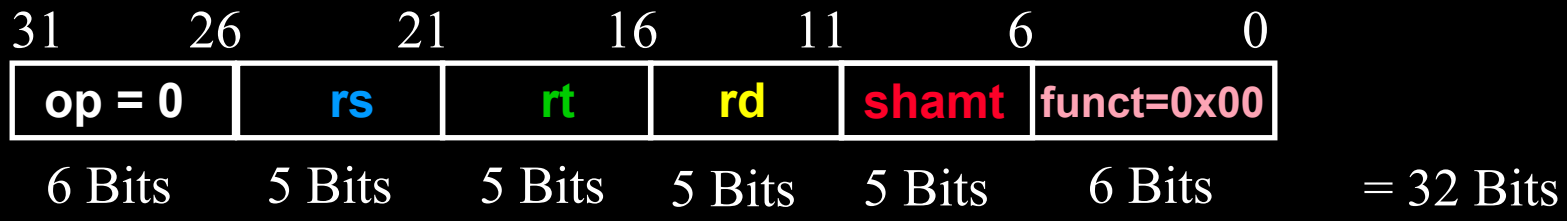
Εντολή SLL (Shift Left Logical)

- **SLL rd, rt, shamt**

IR <= Imem[PC]

reg[rd] <= reg[rt] shifted left logical by shamt

PC <= PC + 4



Η ψευδο-εντολή NOP μετατρέπεται στην εντολή SLL $zero, $zero, 0 με κωδικοποίηση όλα-0

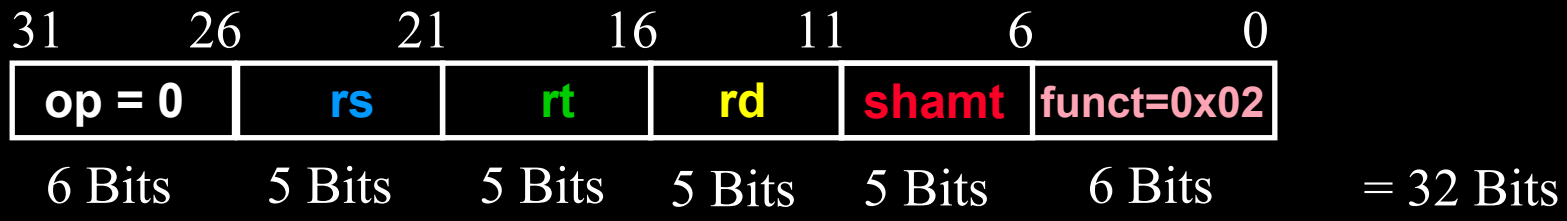
Εντολή SRL (Shift Right Logical)

- SRL rd, rt, shamt

IR <= Imem[PC]

reg[rd] <= reg[rt] shifted right logical by shamt

PC <= PC + 4



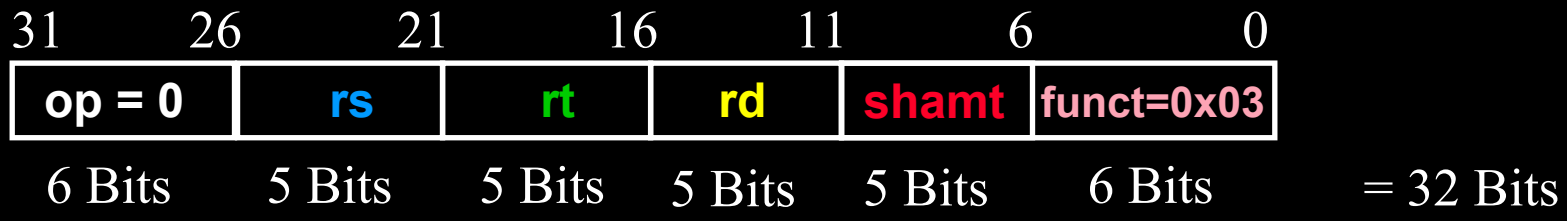
Εντολή SRA (Shift Right Arithmetic)

- SRA rd, rt, shamt

IR <= Imem[PC]

reg[rd] <= reg[rt] shifted right arithmetic by shamt

PC <= PC + 4



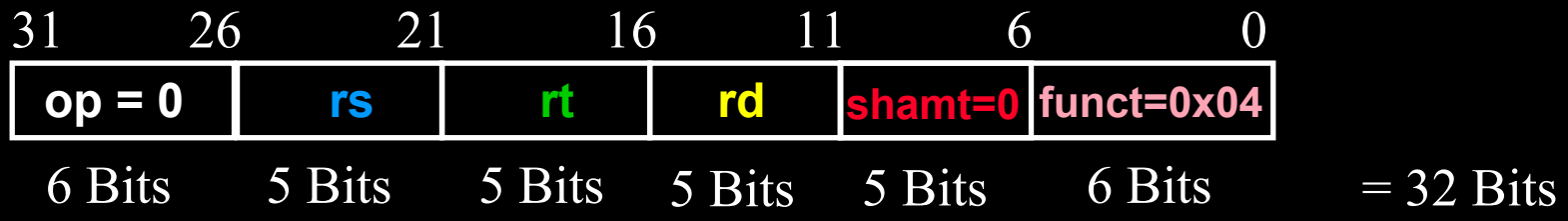
Εντολή SLLV (Shift Left Logical Variable)

- SLLV rd, rt, rs

IR <= Imem[PC]

reg[rd] <= reg[rt] shifted left logical by reg[rs]

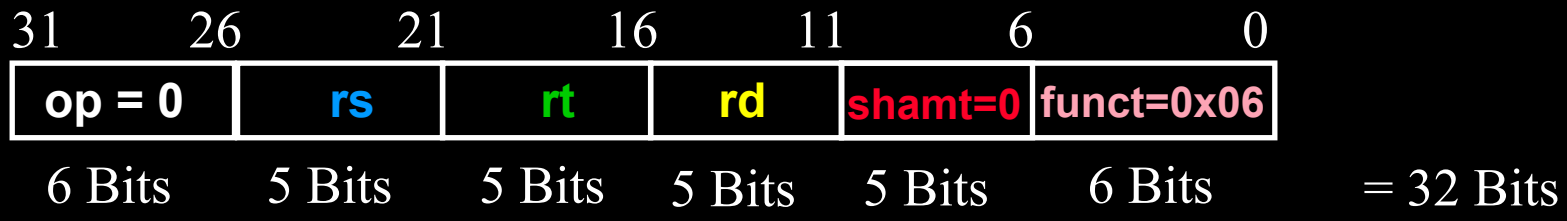
PC <= PC + 4



Εντολή SRLV (Shift Right Logical Variable)

- **SRLV rd, rt, rs**

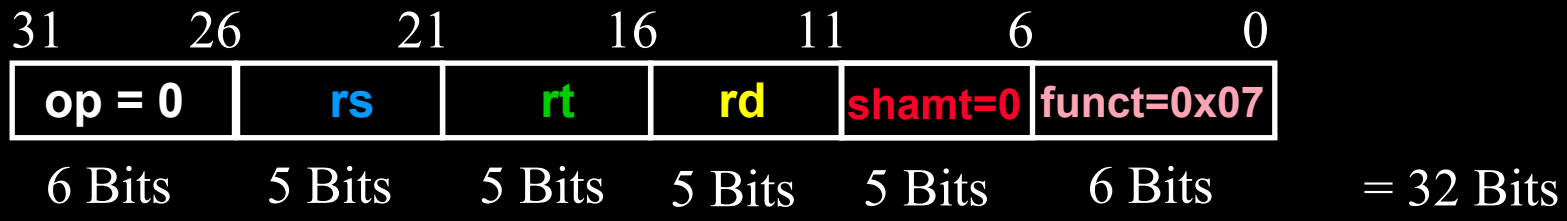
```
IR <= Imem[PC]
reg[rd] <= reg[rt] shifted right logical by reg[rs]
PC <= PC + 4
```



Εντολή SRAV (Shift Right Arithmetic Variable)

- **SRAV rd, rt, rs**

```
IR <= Imem[PC]
reg[rd] <= reg[rt] shifted right arithmetic by reg[rs]
PC <= PC + 4
```



Εντολή SLTI (Set on Less Than Immediate)

- SLTI rt, rs, immediate

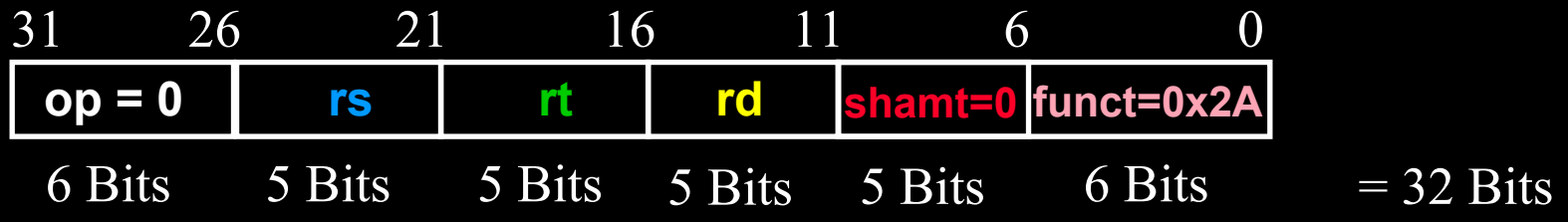
```
IR <= Imem[PC]
IF reg[rs] < sign_extend(immediate)
        THEN reg[rt] <= 1
        ELSE  reg[rt] <= 0
PC <= PC + 4
```

Η σύγκριση γίνεται μεταξύ προσημασμένων ακεραίων αριθμών, χωρίς να γίνεται έλεγχος υπερχείλισης

Εντολή SLT (Set on Less Than)

- SLT rd, rs, rt

```
IR <= Imem[PC]
IF reg[rs] < reg[rt]
        THEN reg[rd] <= 1
        ELSE  reg[rd] <= 0
PC <= PC + 4
```

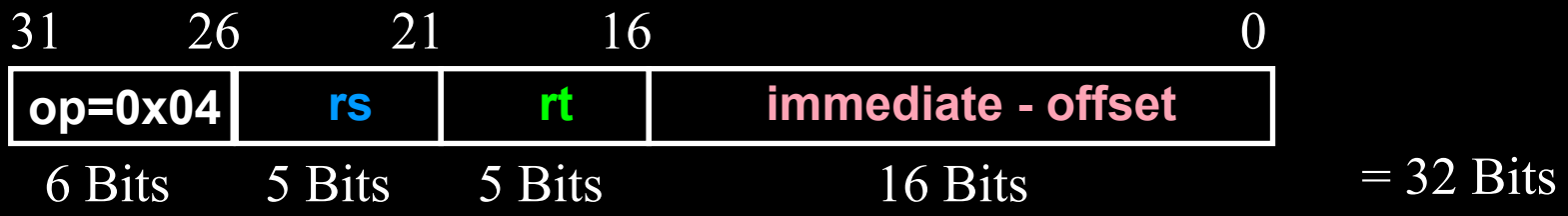


Η σύγκριση γίνεται μεταξύ προσημασμένων ακεραίων αριθμών, χωρίς να γίνεται έλεγχος υπερχείλισης

Εντολή BEQ (Branch on Equal)

- **BEQ rs, rt, immediate (=label)**

```
IR <= Imem[PC]
IF reg[rs] = reg[rt]
        THEN PC <= PC + 4 + 4 x sign_extend(immediate)
        ELSE  PC <= PC + 4
```

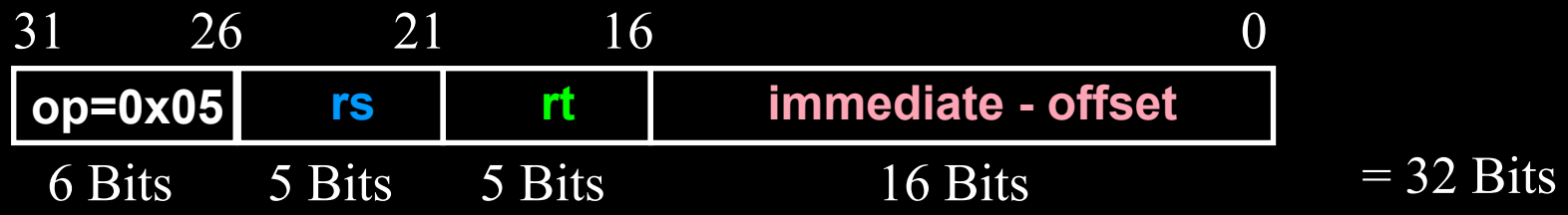


Δεν υλοποιείται η καθυστερημένη διακλάδωση

Εντολή BNE (Branch on Not Equal)

- **BNE rs, rt, immediate (=label)**

```
IR <= Imem[PC]
IF reg[rs] ≠ reg[rt]
        THEN PC <= PC + 4 + 4 x sign_extend(immediate)
        ELSE  PC <= PC + 4
```



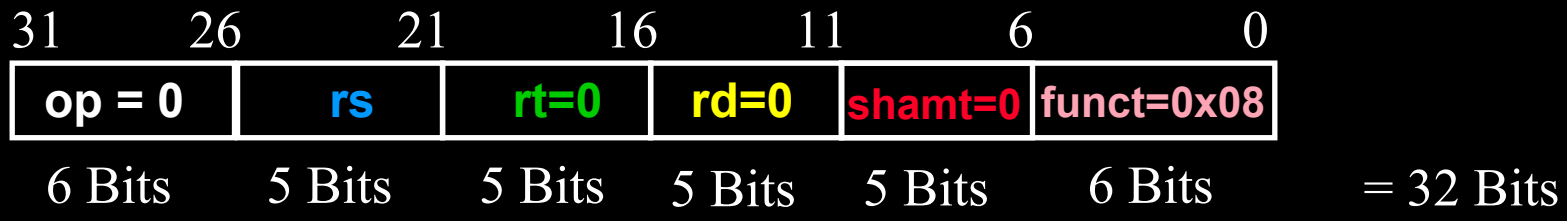
Δεν υλοποιείται η καθυστερημένη διακλάδωση

Εντολή JR (Jump Register)

- **JR rs**

IR <= Imem[PC]

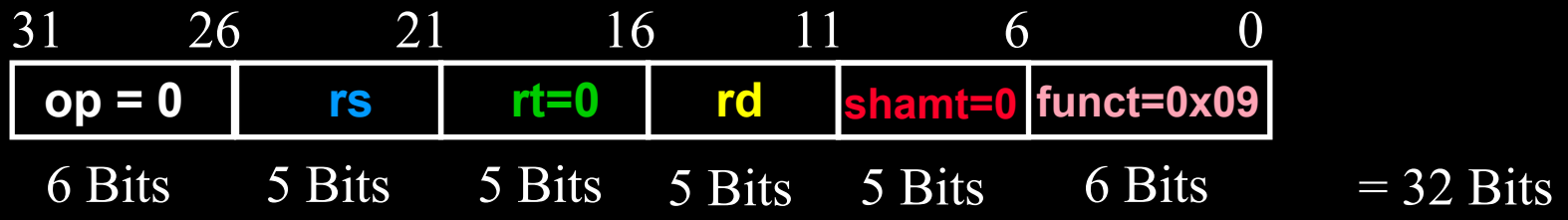
PC <= reg[rs]



Εντολή JALR (Jump And Link Register)

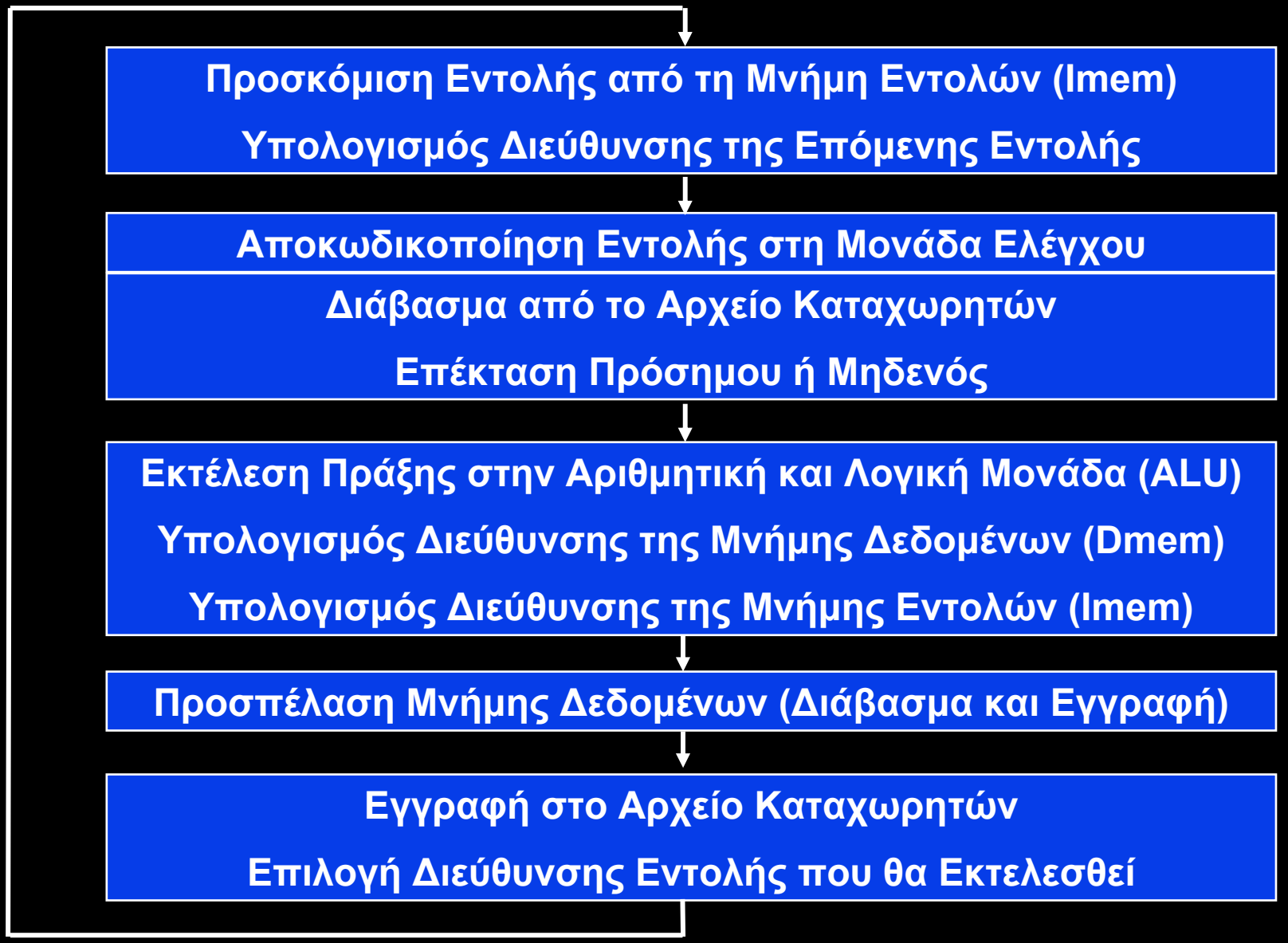
- **JALR rs, rd**

```
IR <= Imem[PC]
reg[rd] <= PC + 4
PC <= reg[rs]
```



Η προκαθορισμένη τιμή (default) του καταχωρητή rd, όταν αυτός δεν δηλώνεται, είναι ο καταχωρητής $ra = $31

Φάσεις Εκτέλεσης Εντολών



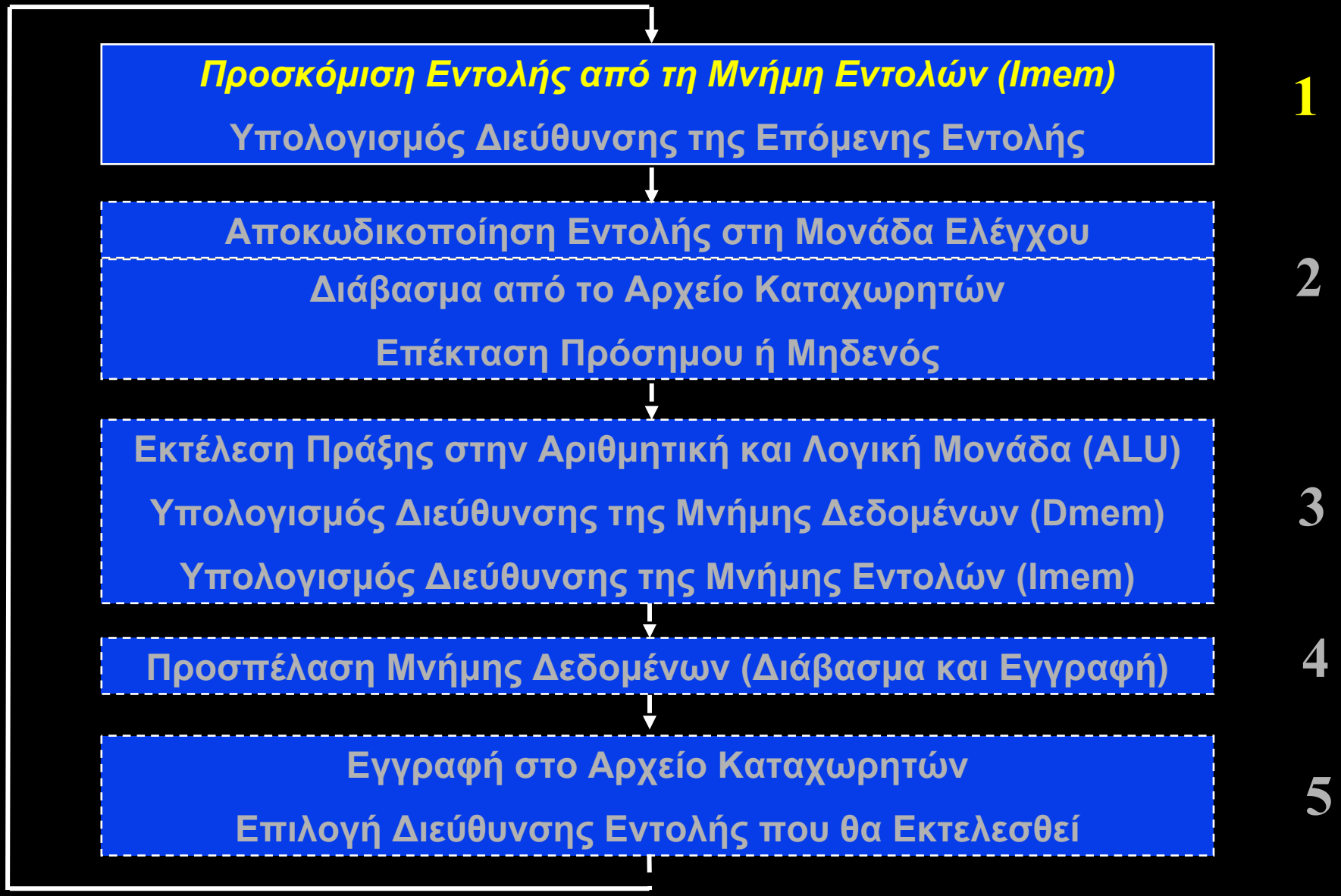
Συσχέτιση Φάσεων με Κύκλους

- Η δίοδος δεδομένων που πρόκειται να σχεδιάσουμε είναι **πολλών κύκλων** με τα ακόλουθα χαρακτηριστικά :
  - Κάθε φάση της εντολής εκτελείται σε **διαφορετικό κύκλο** του ρολογιού
  - Σε έναν κύκλο του ρολογιού μπορεί να μεταφερθούν :
    - δεδομένα από καταχωρητές σε καταχωρητές
    - δεδομένα από καταχωρητή στη μνήμη δεδομένων (εγγραφή)
    - δεδομένα από τη μνήμη δεδομένων σε καταχωρητή (διάβασμα)
    - εντολή από τη μνήμη εντολών σε καταχωρητή (διάβασμα)
  - Για κάθε μεταφορά ορίζουμε μία μικρο-λειτουργία
  - Χρησιμοποιούνται οι καταχωρητές ειδικού σκοπού που έχουμε ήδη ορίσει καθώς και προσωρινοί καταχωρητές, όπου αυτοί απαιτούνται
  - Η εγγραφή στους καταχωρητές γίνεται κατά την επόμενη ακμή του ρολογιού, όταν δεν υπάρχει σήμα ενεργοποίησης εγγραφής
  - Η εγγραφή σε κάποιους καταχωρητές, έστω X, γίνεται κατά την επόμενη ακμή του ρολογιού και όταν το σήμα ελέγχου X_write = 1

Λεπτομερής Ανάλυση των Εντολών του MIPS R2000 σε Μικρο-Λειτουργίες

- Γίνεται σε **χαμηλό** επίπεδο μεταφοράς καταχωρητή, όπου για κάθε μεταφορά ορίζουμε μία μικρο-λειτουργία
- Χρησιμοποιούνται ως καταχωρητές
  - οι καταχωρητές ειδικού σκοπού της διόδου δεδομένων που ήδη έχουμε ορίσει (**PC, IR, MAR, MDR_in, MDR_out**)
  - οι καταχωρητές **reg[rs/rt/rd]** του αρχείου των 32 καταχωρητών, που προσδιορίζονται στα αντίστοιχα πεδία διευθύνσεων καταχωρητών (rs, rt, rd) του IR
  - το πεδίο **immediate** του IR στις εντολές τύπου I
  - το πεδίο **shamt** του IR στις εντολές ολίσθησης με σταθερό αριθμό ολισθήσεων
  - οι **θέσεις μνήμης της μνήμης εντολών** μεγέθους 4 bytes των οποίων η διεύθυνση του περισσότερου σημαντικού byte προσδιορίζεται στο **PC**
  - οι **θέσεις μνήμης της μνήμης δεδομένων** μεγέθους 4 bytes των οποίων η διεύθυνση του περισσότερου σημαντικού byte προσδιορίζεται στο **MAR**
  - **προσωρινοί καταχωρητές, όπου αυτοί απαιτούνται**

Προσκόμιση Εντολής από τη Μνήμη Εντολών



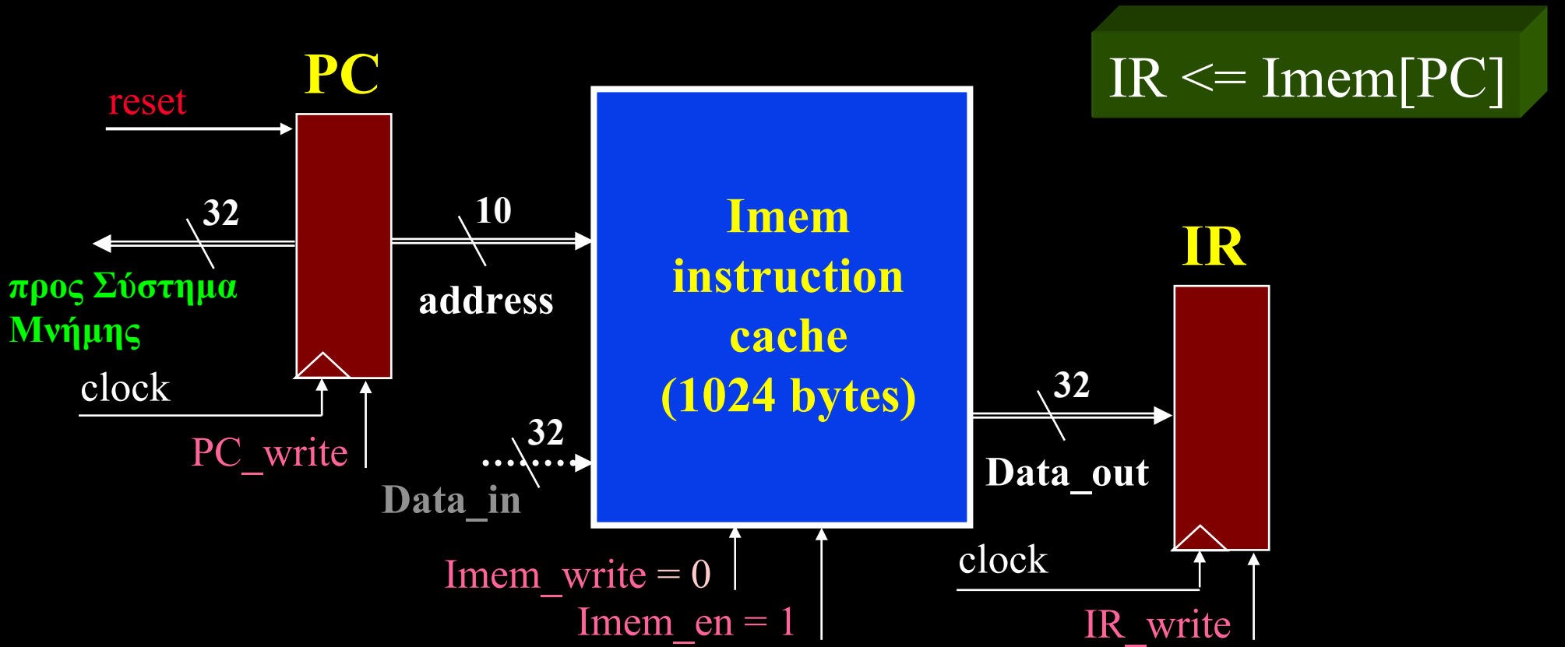
Προσκόμιση Εντολής από τη Μνήμη Εντολών

- Στον καταχωρητή PC (Program Counter) είναι αποθηκευμένη η διεύθυνση της εντολής που πρόκειται να εκτελεσθεί
  - Ο προσδιορισμός της διεύθυνσης της πρώτης εντολής γίνεται κατά τη φάση αρχικοποίησης του επεξεργαστή με το σήμα reset
  - Ο προσδιορισμός της διεύθυνσης της επόμενης εντολής γίνεται κατά την τελευταία φάση εκτέλεσης της τρέχουσας εντολής
- Από τη μνήμη εντολών Imem (Instruction Memory) διαβάζεται η εντολή που πρόκειται να εκτελεσθεί, όταν Imem_en = 1 και Imem_write = 0
- Στον καταχωρητή IR (Instruction Register) αποθηκεύεται η εντολή που πρόκειται να εκτελεσθεί κατά την επόμενη ακμή του ρολογιού και όταν το σήμα ελέγχου IR_write = 1

Είναι κοινή διαδικασία για όλες τις εντολές

IR <= Imem[PC]

Προσκόμιση Εντολής από τη Μνήμη Εντολών

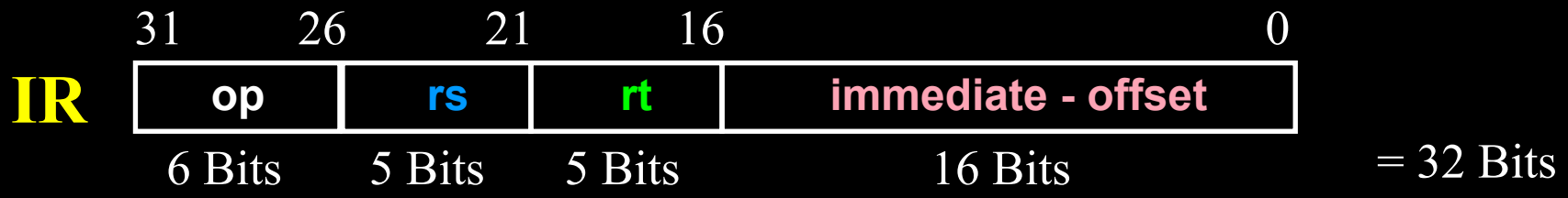


Αναφερόμαστε μόνο στη διαδικασία προσκόμισης εντολών από τη μνήμη εντολών

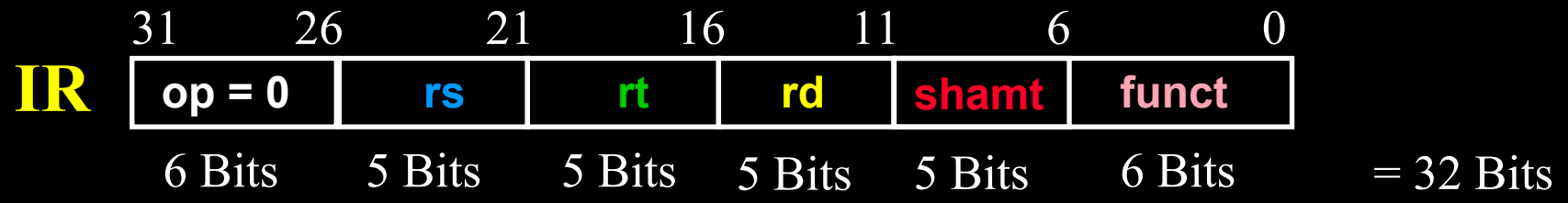
Τα 2 λιγότερο σημαντικά ψηφία της διεύθυνσης είναι πάντα 0 λόγω ευθυγράμμισης

Προσκόμιση Εντολής από τη Μνήμη Εντολών

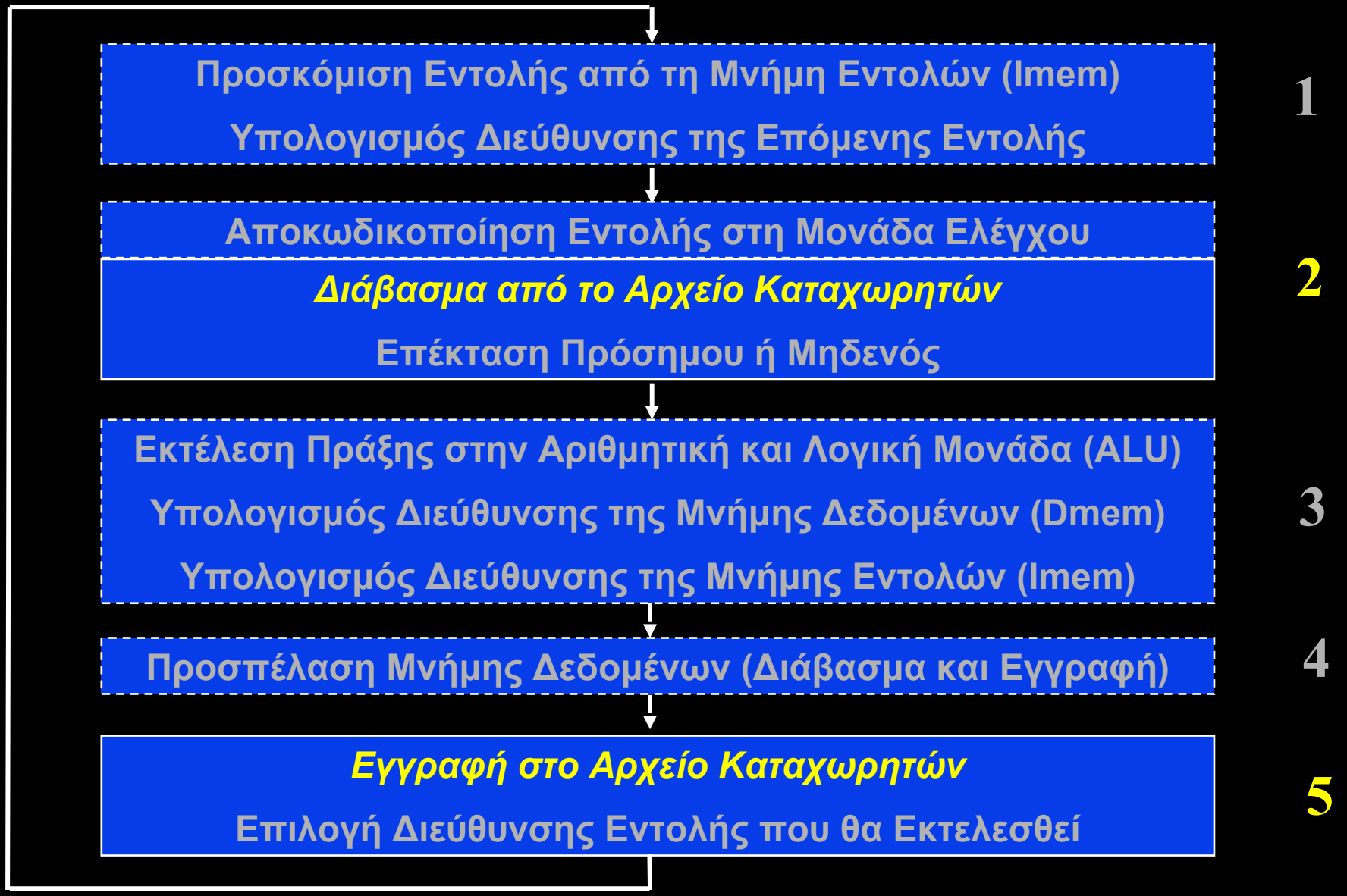
- Μετά την προσκόμιση της εντολής το περιεχόμενο του καταχωρητή IR (Instruction Register) είναι:
  - για τις εντολές τύπου Immediate



  - για τις εντολές τύπου Register



Προσπέλαση του Αρχείου Καταχωρητών



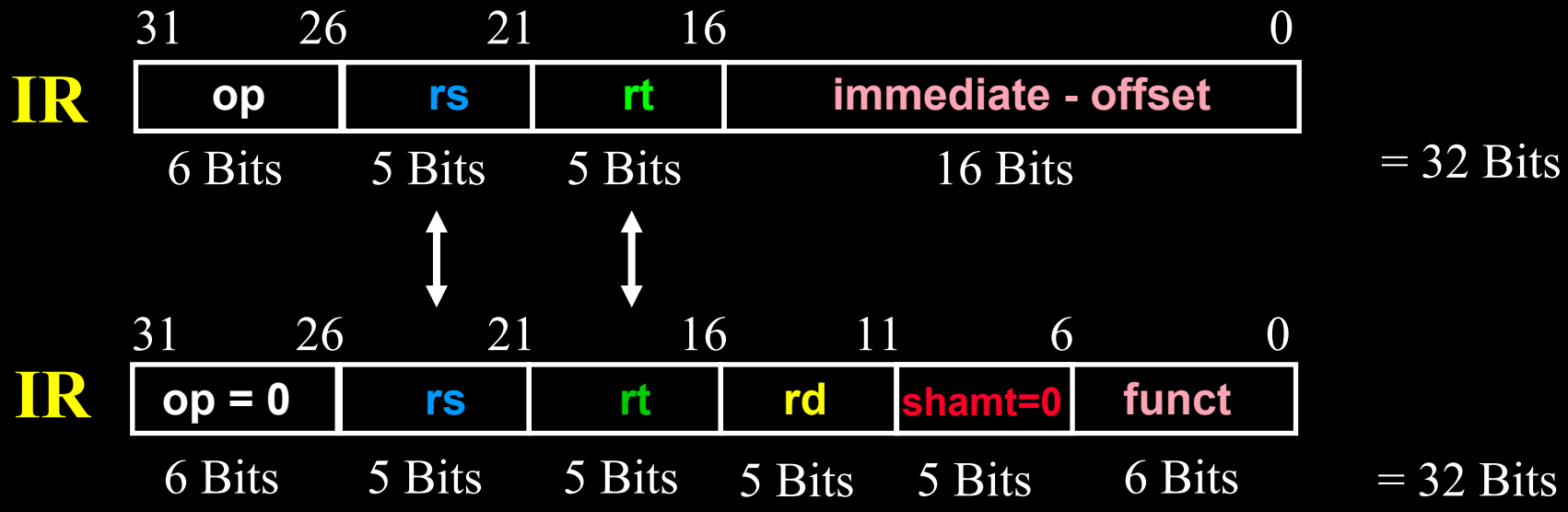
Διάβασμα από το Αρχείο Καταχωρητών

- **Διάβασμα από τον καταχωρητή rs**
  - reg[rs] = διεύθυνση της μνήμης δεδομένων (LW, SW)
  - reg[rs] = ο πρώτος από τους δύο τελεστέους (ADDIU*, ADDU*, SLTI, SLT, BEQ, BNE)
  - reg[rs] = διεύθυνση της μνήμης εντολών (JR, JALR)
  - reg[rs] = ο αριθμός των ολισθήσεων (SLLV, SRLV, SRAV)
- **Διάβασμα από τον καταχωρητή rt**
  - reg[rt] = ο τελεστέος που θα αποθηκευθεί στη μνήμη δεδομένων (SW)
  - reg[rt] = ο δεύτερος από τους δύο τελεστέους (ADDU*, SLT, BEQ, BNE)
  - reg[rt] = ο τελεστέος που θα υποστεί ολίσθηση (εντολές ολίσθησης)

* και οι παρόμοιες αριθμητικές και λογικές εντολές

Διάβασμα από το Αρχείο Καταχωρητών

- **Δύο διαβάσματα παράλληλα από rs και rt**
  - **Τα πεδία rs και rt είναι κοινά και για τους δύο τύπους εντολών I και R**



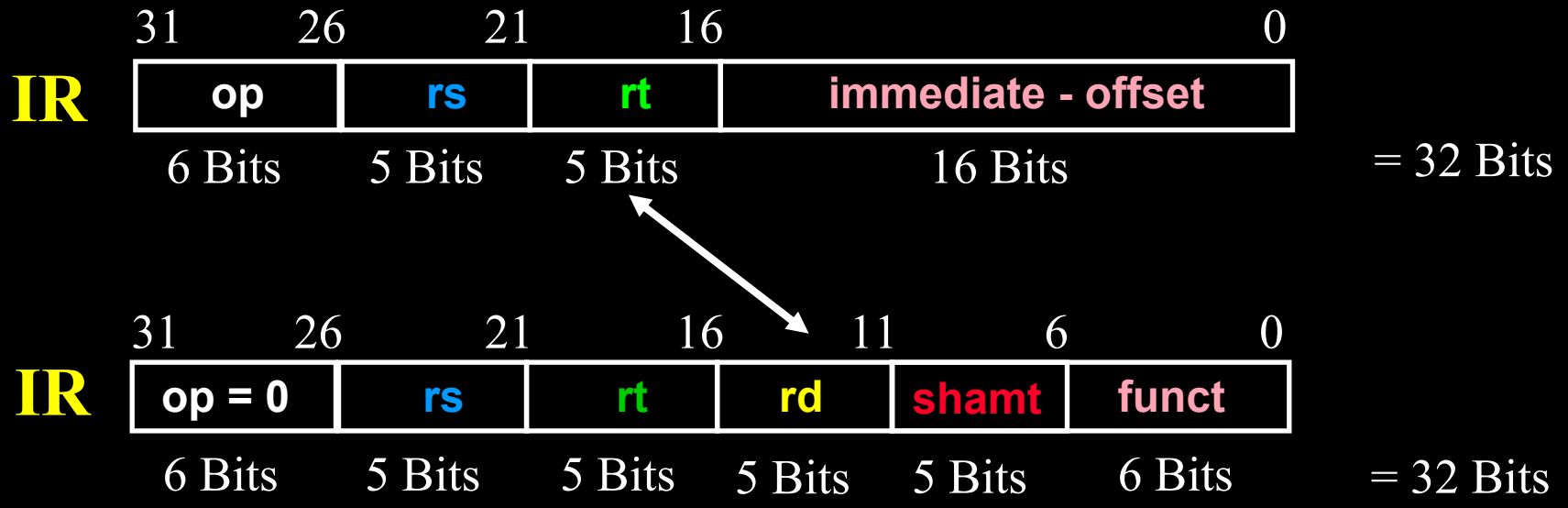
Εγγραφή στο Αρχείο Καταχωρητών

- **Εγγραφή στον καταχωρητή rt**
  - reg[rt] = ο τελεστέος που θα φορτωθεί από τη μνήμη δεδομένων (LW)
  - reg[rt] = το αποτέλεσμα της αριθμητικής ή λογικής πράξης (ADDIU*)
  - reg[rt] = το αποτέλεσμα της σύγκρισης (0 ή 1) (SLTI)
- **Εγγραφή στον καταχωρητή rd**
  - reg[rd] = το αποτέλεσμα της αριθμητικής ή λογικής πράξης (ADDU*)
  - reg[rd] = το αποτέλεσμα της ολίσθησης (εντολές ολίσθησης)
  - reg[rd] = το αποτέλεσμα της σύγκρισης (0 ή 1) (SLT)
  - reg[rd] = η διεύθυνση της επόμενης εντολής (PC+4) (JALR)

* και οι παρόμοιες αριθμητικές και λογικές εντολές

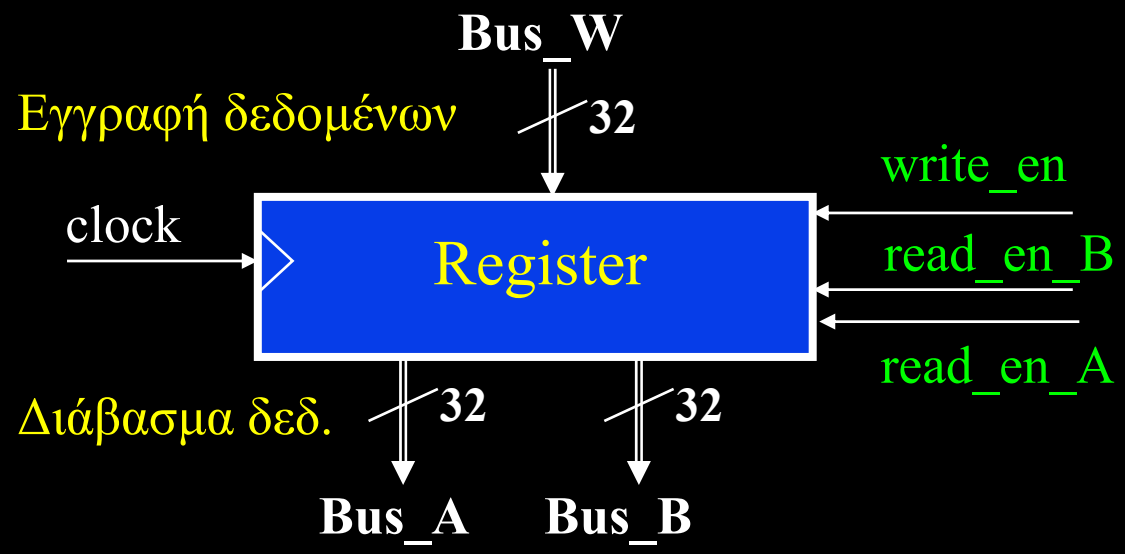
Εγγραφή στο Αρχείο Καταχωρητών

- **Μία εγγραφή στον rt για εντολές τύπου I ή στον rd για εντολές τύπου R**
  - **Τα πεδία rt και rd είναι διαφορετικά στους δύο τύπους εντολών I και R, αντίστοιχα**

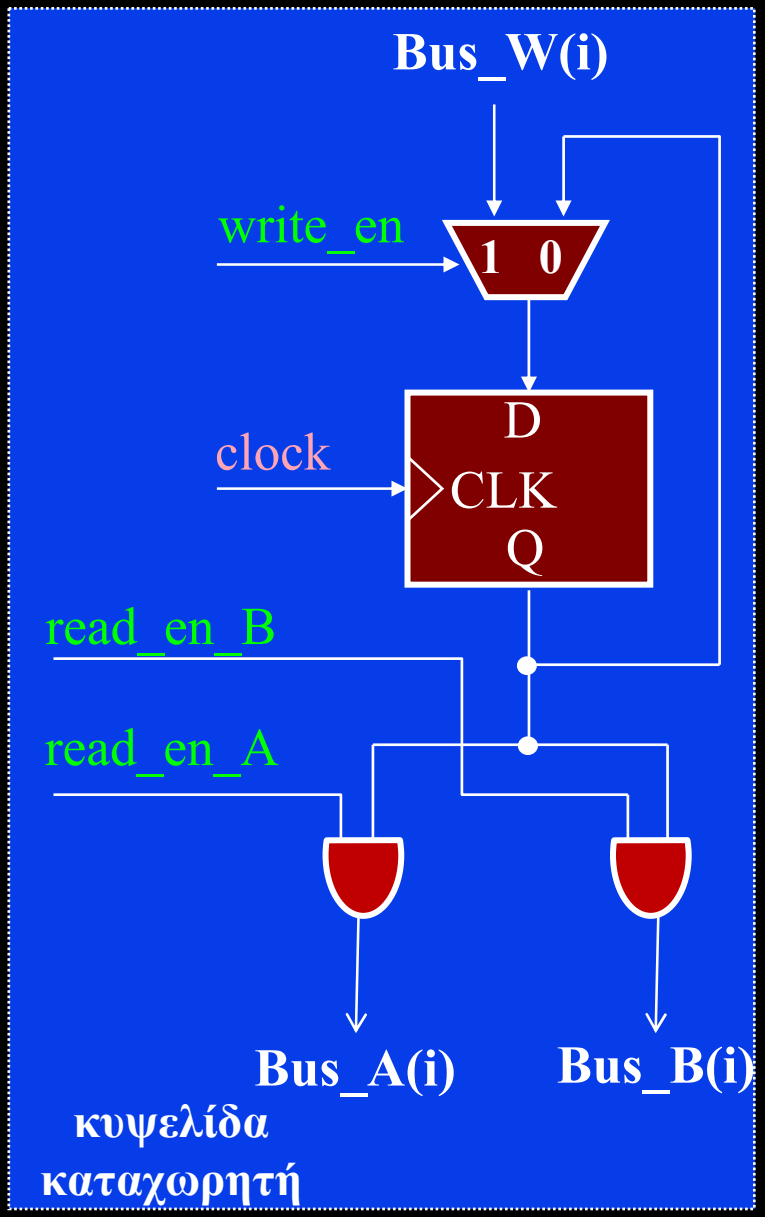


Καταχωρητές Γενικής Χρήσης

- **Με μία αρτηρία εγγραφής Bus_W και δύο αρτηρίες διαβάσματος Bus_A και Bus_B**



write_en = 0 : δεν αλλάζει δεδομένα
write_en = 1 : εγγραφή νέων δεδομένων από Bus_W
read_en_A = 0 : έξοδος "0"
read_en_A = 1 : διάβασμα δεδομένων στο Bus_A
read_en_B = 0 : έξοδος "0"
read_en_B = 1 : διάβασμα δεδομένων στο Bus_B



Διάβασμα από το Αρχείο Καταχωρητών

- Υποστηρίζει δύο αρτηρίες διαβάσματος (Bus_A & Bus_B)
  - Η αρτηρία επιλογής Select_A συνδέεται με το πεδίο rs του καταχωρητή IR και επιλέγει τον καταχωρητή rs, που θα διαβαστεί και θα εμφανίσει το περιεχόμενό του στο Bus_A, ασύγχρονα
  - Στην επόμενη ακμή του ρολογίου αποθηκεύεται στον προσωρινό καταχωρητή A το περιεχόμενο του καταχωρητή rs που μεταφέρεται μέσω του Bus_A
  - Η αρτηρία επιλογής Select_B συνδέεται με το πεδίο rt του καταχωρητή IR και επιλέγει τον καταχωρητή rt, που θα διαβαστεί και θα εμφανίσει το περιεχόμενό του στο Bus_B, ασύγχρονα
  - Στην επόμενη ακμή του ρολογίου αποθηκεύεται στον προσωρινό καταχωρητή B το περιεχόμενο του καταχωρητή rt που μεταφέρεται μέσω του Bus_B

A <= reg[rs]

B <= reg[rt]

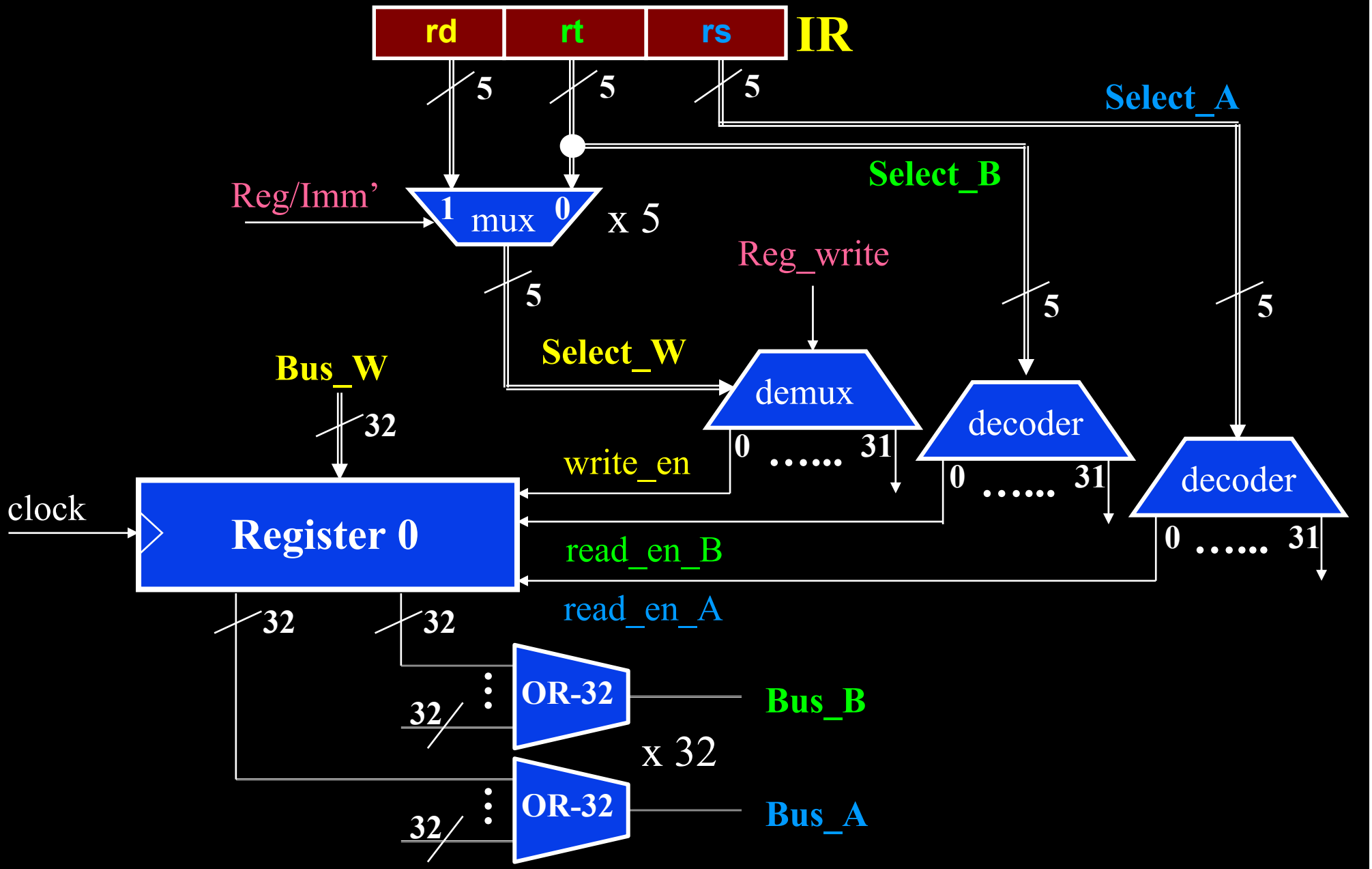
Εγγραφή στο Αρχείο Καταχωρητών

- Υποστηρίζει μία αρτηρία εγγραφής (Bus_W)
  - Η αρτηρία επιλογής Select_W συνδέεται με το πεδίο rt του καταχωρητή IR για τις εντολές τύπου I, όταν το σήμα ελέγχου Reg/Imm' = 0
  - Η αρτηρία επιλογής Select_W συνδέεται με το πεδίο rd του καταχωρητή IR για τις εντολές τύπου R, όταν το σήμα ελέγχου Reg/Imm' = 1
  - Η αρτηρία επιλογής Select_W επιλέγει αντίστοιχα τον καταχωρητή rt ή rd που θα γραφτεί με νέα δεδομένα, που μεταφέρονται μέσω του Bus_W, κατά την επόμενη ακμή του ρολογιού και όταν το σήμα ελέγχου Reg_write = 1
  - Η σύνδεση της αρτηρίας επιλογής Select_W με τον καταχωρητή IR γίνεται μέσω ενός 5-ψήφιου πολυπλέκτη 2 σε 1 με σήμα επιλογής το σήμα ελέγχου Reg/Imm'

reg[rt/rd] <= Bus_W

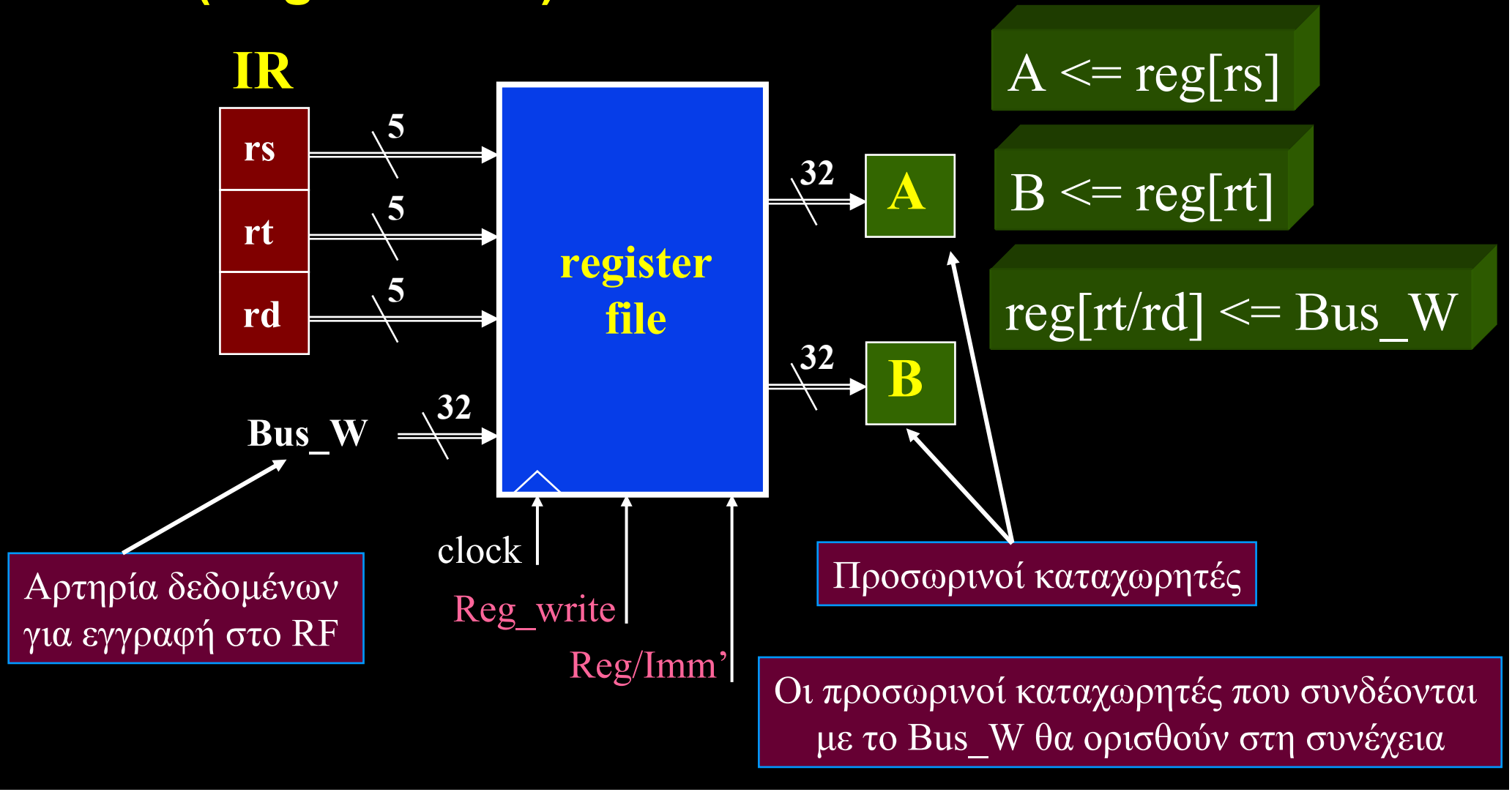
Οι προσωρινοί καταχωρητές που συνδέονται με το Bus_W θα ορισθούν στη συνέχεια

Αρχείο Καταχωρητών

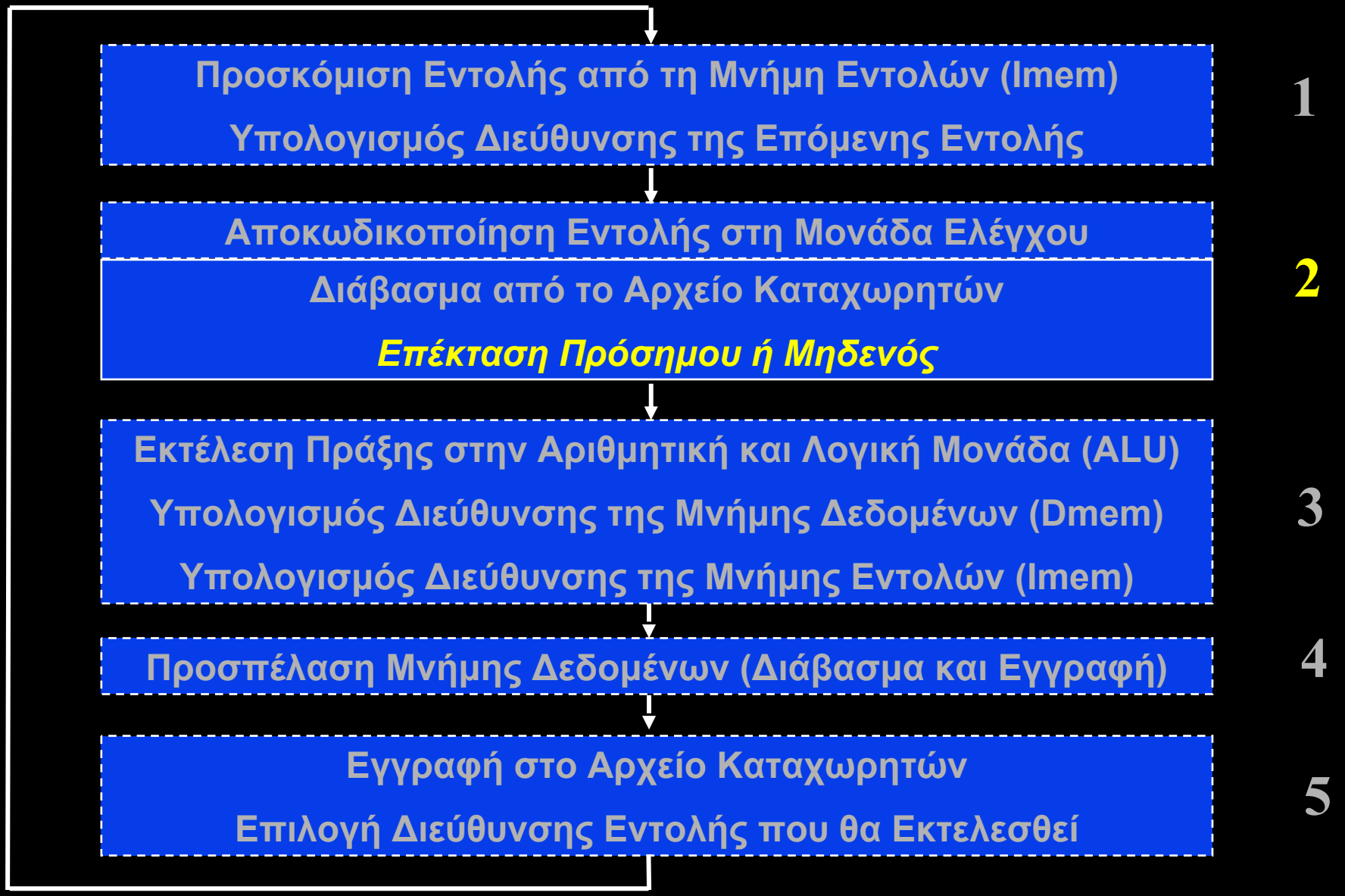


Αρχείο Καταχωρητών

- **Αρχείο 32 Καταχωρητών των 32 ψηφίων (Register File)**



Επέκταση Πρόσημου ή Μηδενός



Κύκλωμα Επέκτασης Πεδίου Immediate

- Το πεδίο **immediate** του καταχωρητή IR έχει μέγεθος **16 ψηφία**, ενώ οι λειτουργικές μονάδες του επεξεργαστή επεξεργάζονται δεδομένα μεγέθους **32 ψηφιών**
- Το πεδίο **immediate** του καταχωρητή IR αποθηκεύεται στον **προσωρινό καταχωρητή I**, αφού υποστεί :
  - **επέκταση πρόσημου** από 16 ψηφία σε 32 ψηφία, όταν το σήμα ελέγχου Sign/Zero' = 1

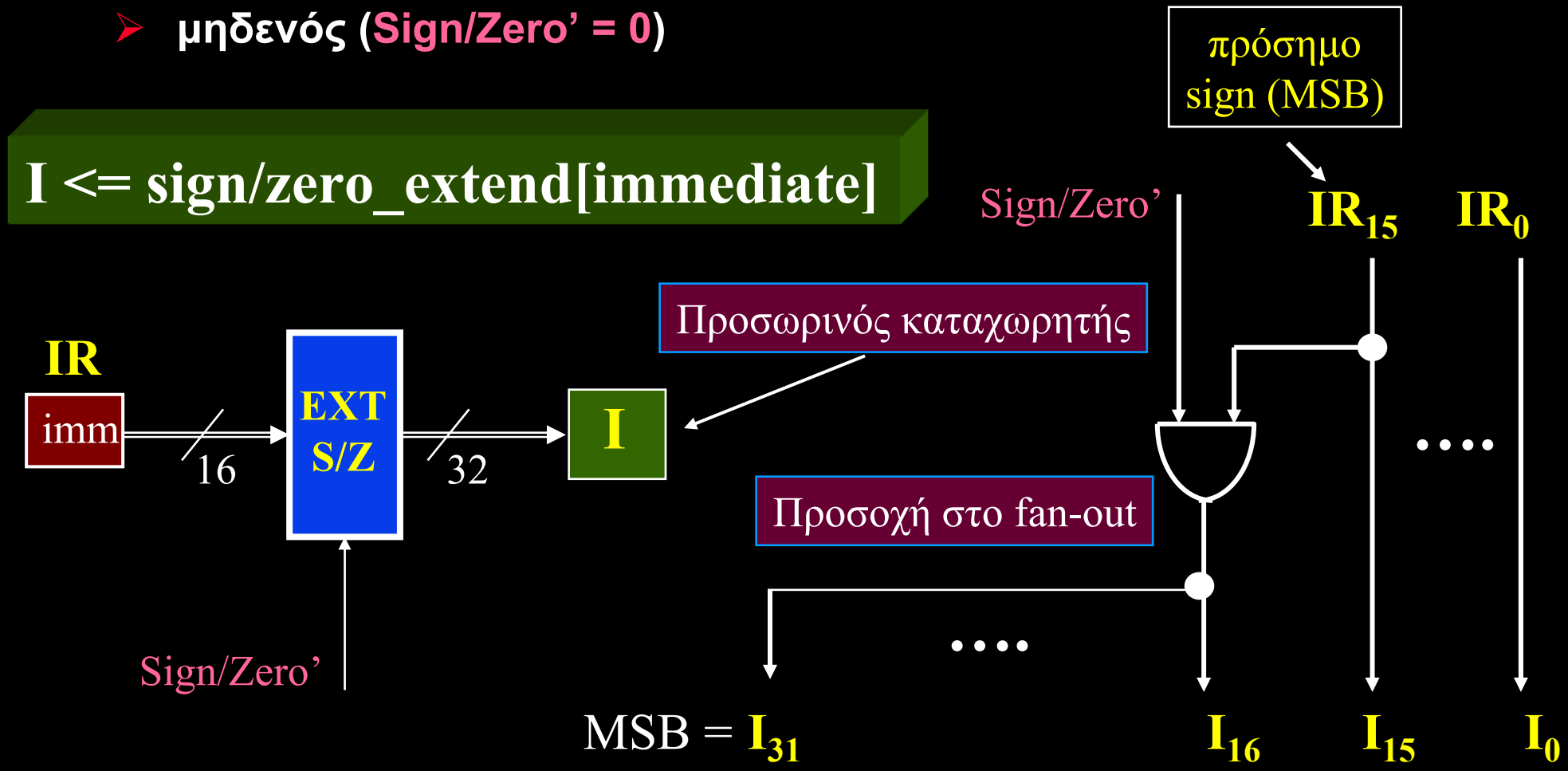
    (τα 16 περισσότερα σημαντικά ψηφία που προστίθενται ακολουθούν το πρόσημο)
  - **επέκταση μηδενός** από 16 ψηφία σε 32 ψηφία, όταν το σήμα ελέγχου Sign/Zero' = 0

    (τα 16 περισσότερα σημαντικά ψηφία που προστίθενται είναι 0)

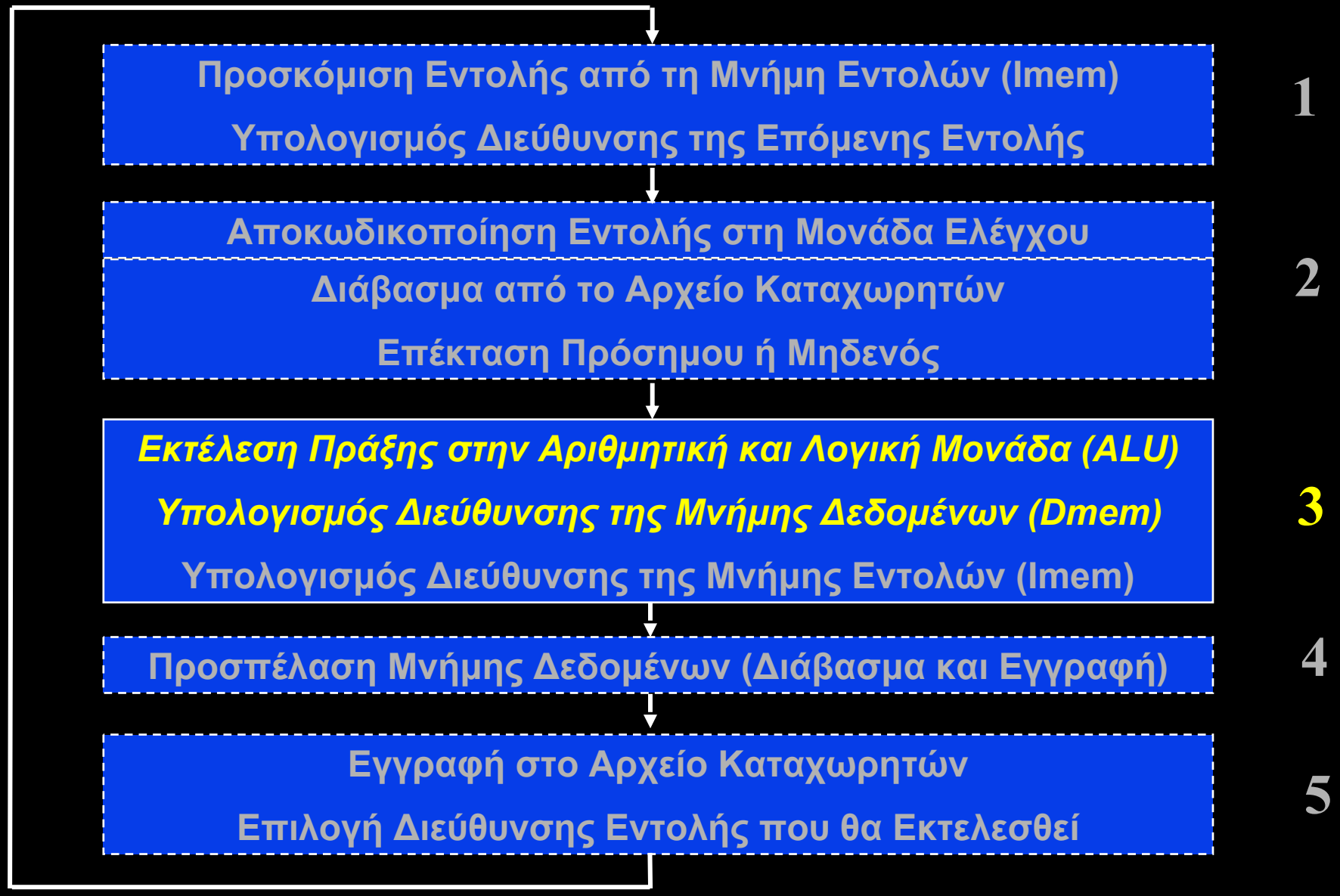
I <= sign/zero_extend[immediate]

Κύκλωμα Επέκτασης Πεδίου Immediate

- **Κύκλωμα επέκτασης πεδίου immediate**
  - πρόσημου (Sign/Zero' = 1)
  - μηδενός (Sign/Zero' = 0)



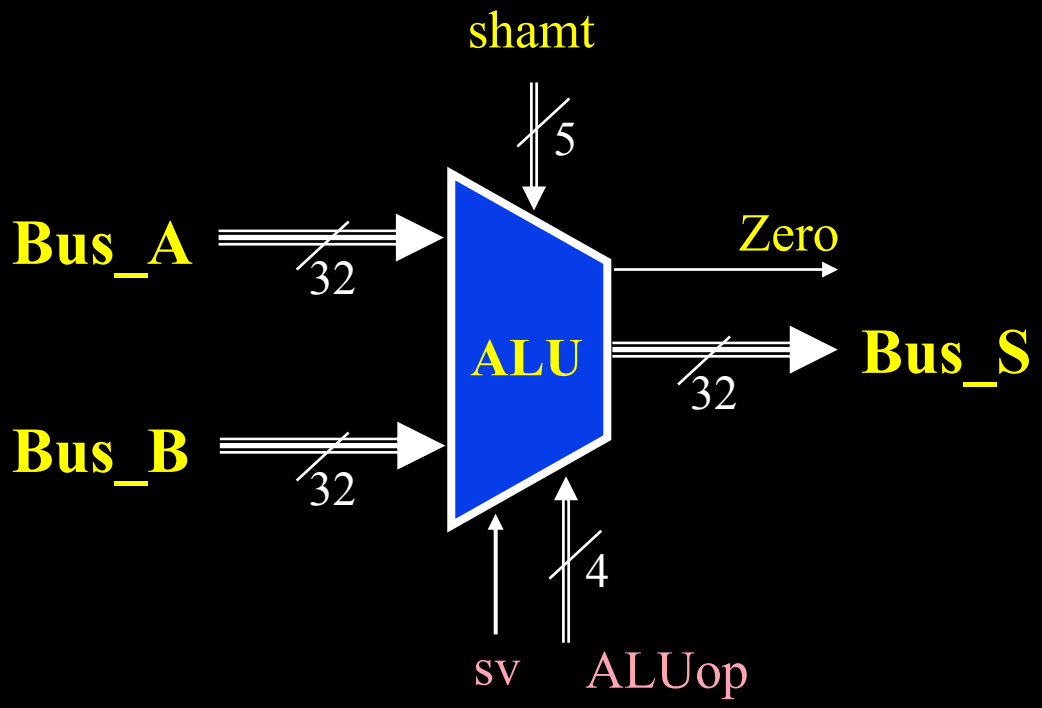
Αριθμητική και Λογική Μονάδα



Αριθμητική και Λογική Μονάδα (ALU)

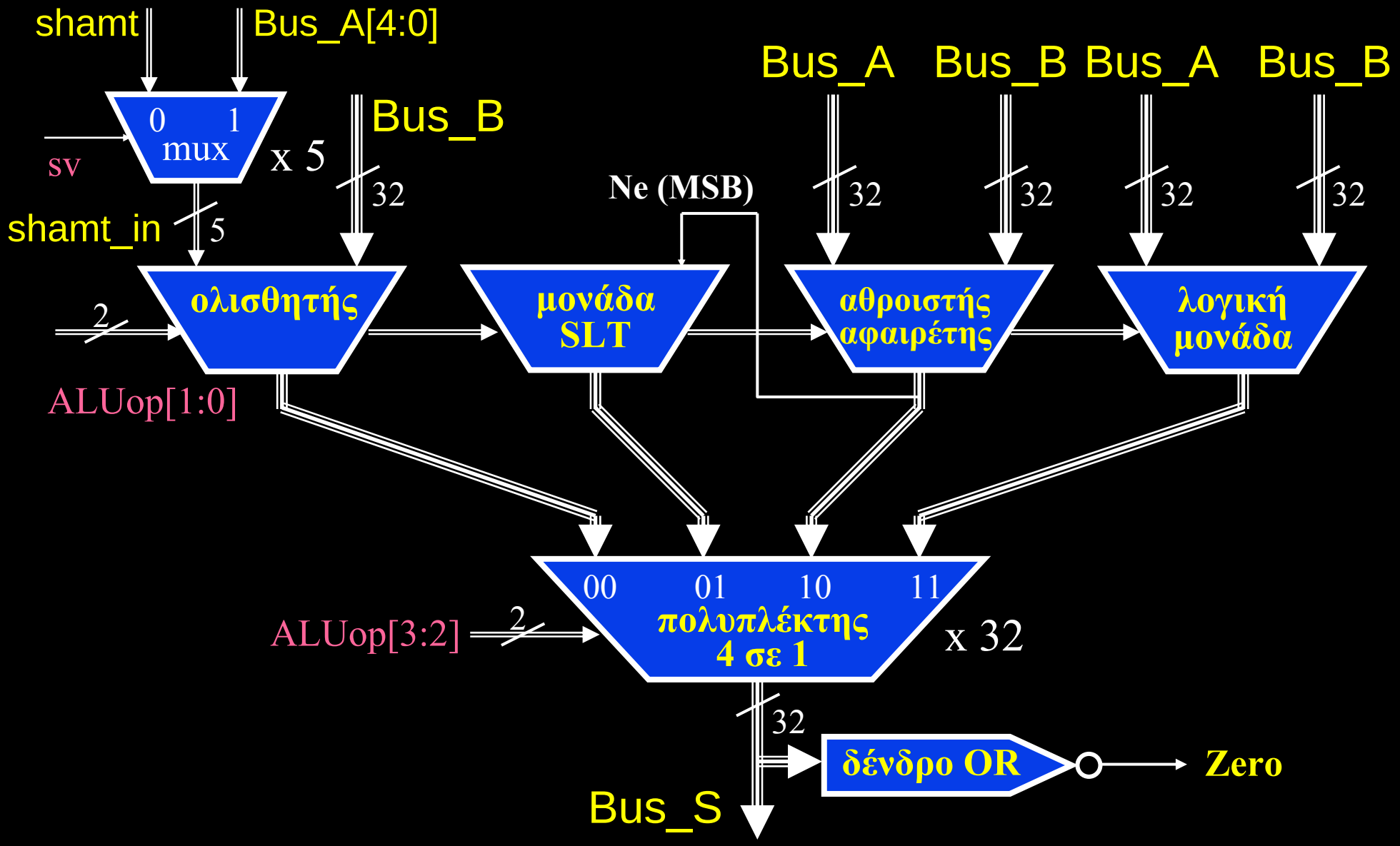
- **Εκτέλεση αριθμητικών και λογικών πράξεων**
  - πρόσθεση προσημασμένων ακεραίων αριθμών
    (κατά την εκτέλεση των εντολών: LW, SW, ADDIU, ADDU)
  - αφαίρεση προσημασμένων ακεραίων αριθμών
    (κατά την εκτέλεση των εντολών: SUBU, SLTI, SLT, BEQ, BNE)
  - λογικό AND, OR, XOR, NOR ανά ψηφίο
    (κατά την εκτέλεση των εντολών: ANDI, ORI, XORI, AND, OR, XOR, NOR)
- **Εκτέλεση ολισθήσεων (με τη χρήση ολισθητή)**
  - κατά την εκτέλεση των εντολών SLL, SRL, SRA, SLLV, SRLV, SRAV
- **Ανίχνευση μηδενός στο αποτέλεσμα της αφαίρεσης (με τη χρήση δένδρου πυλών OR)**
  - κατά την εκτέλεση των εντολών BEQ και BNE
- **Εμφάνιση του πρόσημου του αποτελέσματος της αφαίρεσης στην έξοδο της ALU**
  - κατά την εκτέλεση των εντολών SLTI και SLT

Αριθμητική και Λογική Μονάδα (ALU)



Bus_A, Bus_B, Bus_S : οι δύο είσοδοι και η έξοδος της ALU μεγέθους 32 ψηφίων
shamt : η αρτηρία μεγέθους 5 ψηφίων που προσδιορίζει το σταθερό αριθμό ολισθήσεων
ALUop : το σήμα ελέγχου μεγέθους 4 ψηφίων που επιλέγει την πράξη
sv : το σήμα ελέγχου που μας πληροφορεί ότι ο αριθμός των ολισθήσεων είναι μεταβλητός
Zero : το σήμα κατάστασης που μας πληροφορεί ότι η έξοδος της ALU (Bus_S) είναι μηδέν

Αριθμητική και Λογική Μονάδα (ALU)



Σήματα Ελέγχου της ALU

εντολή	πεδίο funct		ALUop	sv
ADDU	33	1000 01	10 01	X
SUBU	35	1000 11	10 11	X
AND	36	1001 00	11 00	X
OR	37	1001 01	11 01	X
XOR	38	1001 10	11 10	X
NOR	39	1001 11	11 11	X
SLL	0	0000 00	00 00	0
SRL	2	0000 10	00 10	0
SRA	3	0000 11	00 11	0
SLLV	4	0001 00	00 00	1
SRLV	6	0001 10	00 10	1
SRAV	7	0001 11	00 11	1
SLT	42	1010 10	01 10	X

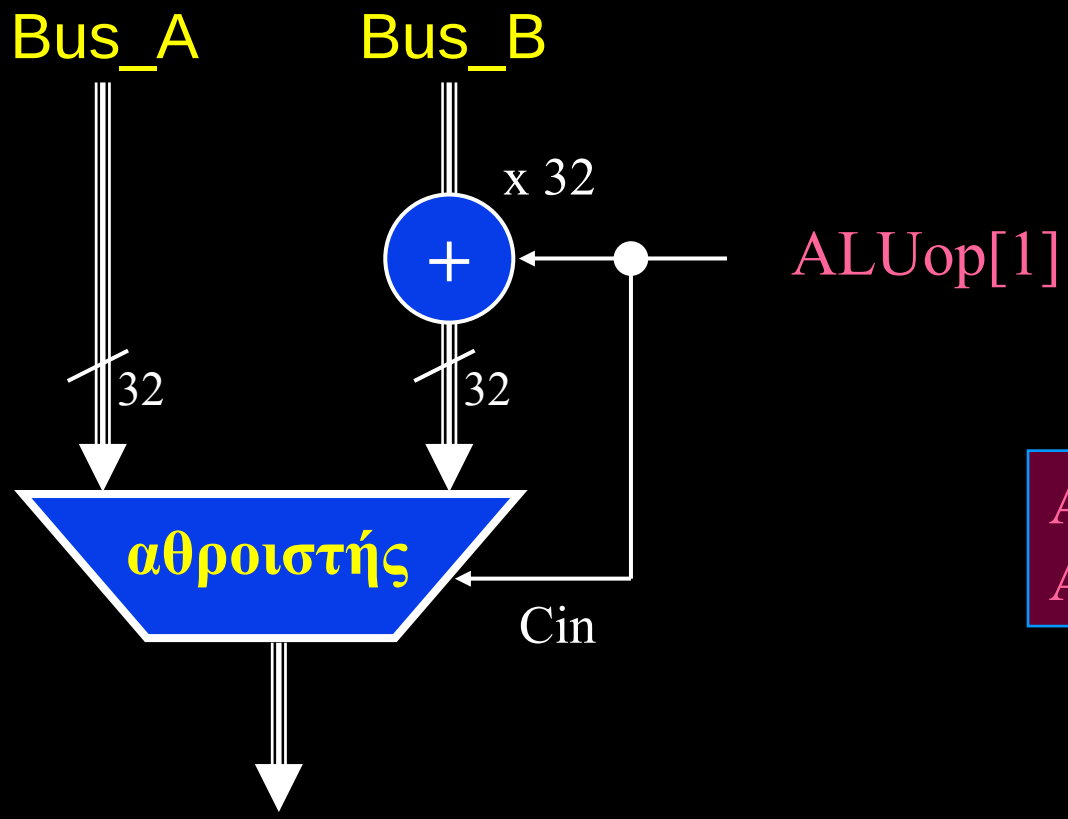
funct[1:0] = ALUop[1:0]

funct[2] = sv

Σήματα Ελέγχου της ALU

εντολή	πεδίο opcode		ALUop
ADDIU	9	001001	1001
ANDI	12	001100	1100
ORI	13	001101	1101
XORI	14	001110	1110
LW	35	100011	10 01
SW	43	101011	10 01
BEQ	4	000100	1011
BNE	5	000101	1011
SLTI	10	001010	0110

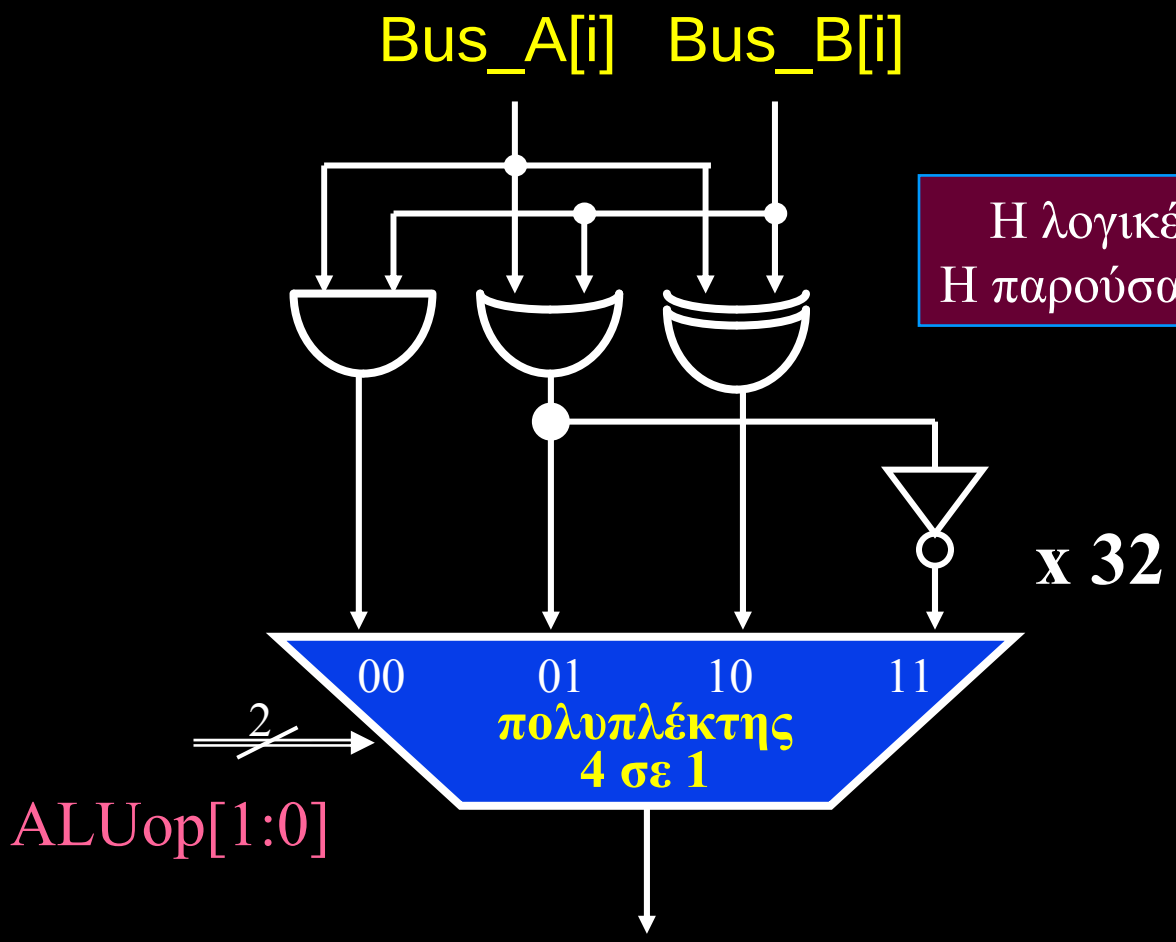
Ο Αθροιστής - Αφαιρέτης της ALU



ALU0p[1] = 0: Πρόσθεση
ALU0p[1] = 1: Αφαίρεση

Bus_A, Bus_B : οι δύο είσοδοι του Αθροιστή - Αφαιρέτη μεγέθους 32 ψηφίων
ALUop[1] : το σήμα ελέγχου που επιλέγει την πράξη (πρόσθεση ή αφαίρεση)
Cin : το κρατούμενο εισόδου
Το κρατούμενο εξόδου δεν χρησιμοποιείται

Η Λογική Μονάδα της ALU

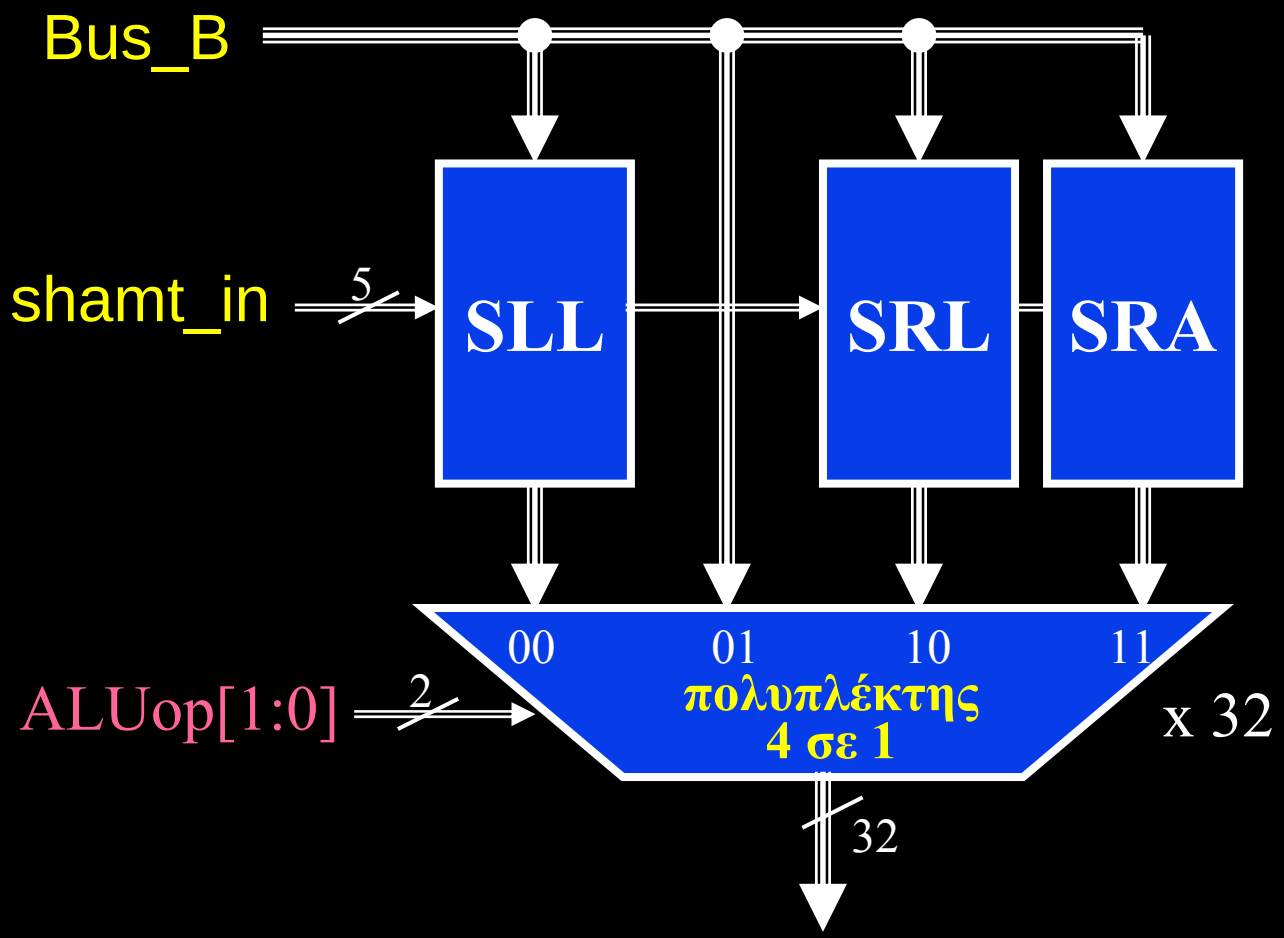


Η λογικές πράξεις εκτελούνται ανά ψηφίο.
Η παρούσα μονάδα επαναλαμβάνεται 32 φορές

ALUop[1:0] = 00: AND
ALUop[1:0] = 01: OR
ALUop[1:0] = 10: XOR
ALUop[1:0] = 11: NOR

Bus_A, Bus_B : οι δύο είσοδοι της Λογικής Μονάδας μεγέθους 32 ψηφίων
ALUop[1:0] : το σήμα ελέγχου μεγέθους 2 ψηφίων που επιλέγει τη λογική πράξη

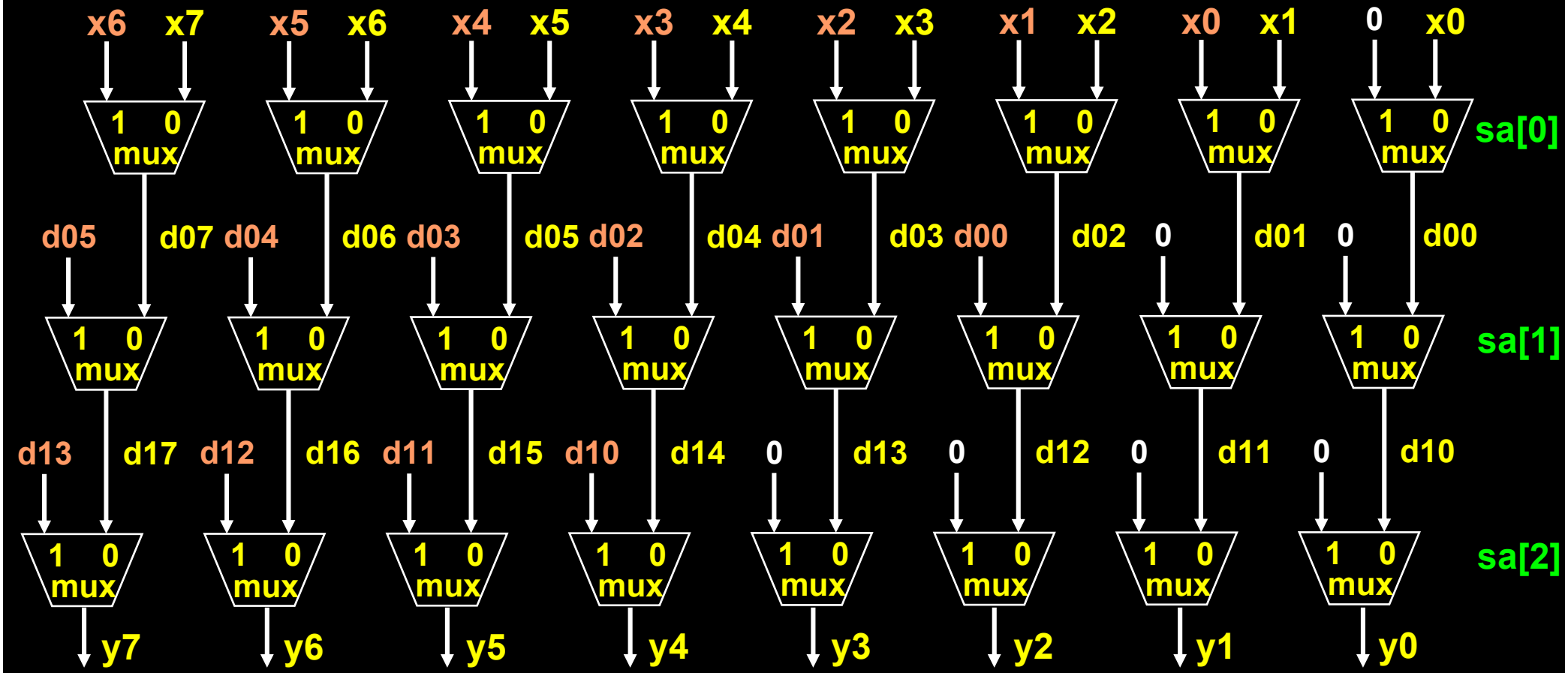
Ο Ολισθητής της ALU



ALUop[1:0] = 00: SLL
ALUop[1:0] = 01: τίποτα
ALUop[1:0] = 10: SRL
ALUop[1:0] = 11: SRA

Bus_B : η είσοδος του Ολισθητή μεγέθους 32 ψηφίων
shamt_in : η αρτηρία μεγέθους 5 ψηφίων που προσδιορίζει τον αριθμό των ολισθήσεων
ALUop[1:0] : το σήμα ελέγχου μεγέθους 2 ψηφίων που επιλέγει την πράξη της ολίσθησης
ALUop[1:0]=01 : η είσοδος του Ολισθητή οδηγείται στην έξοδο χωρίς να υποστεί ολίσθηση

8 - Ψήφιος Ολισθητής Αριστερής Ολίσθησης (Shifter for Shift Left Logical)



Κάθε επίπεδο πολυπλεκτών i εκτελεί ολίσθηση κατά την απόσταση 2^i και όλοι οι πολυπλέκτες 2 σε 1 αυτού του επιπέδου λαμβάνουν σαν είσοδο επιλογής το σήμα sa[i]

sa[2:0] : η αρτηρία μεγέθους 3 ψηφίων που προσδιορίζει τον αριθμό των ολισθήσεων (0..7)

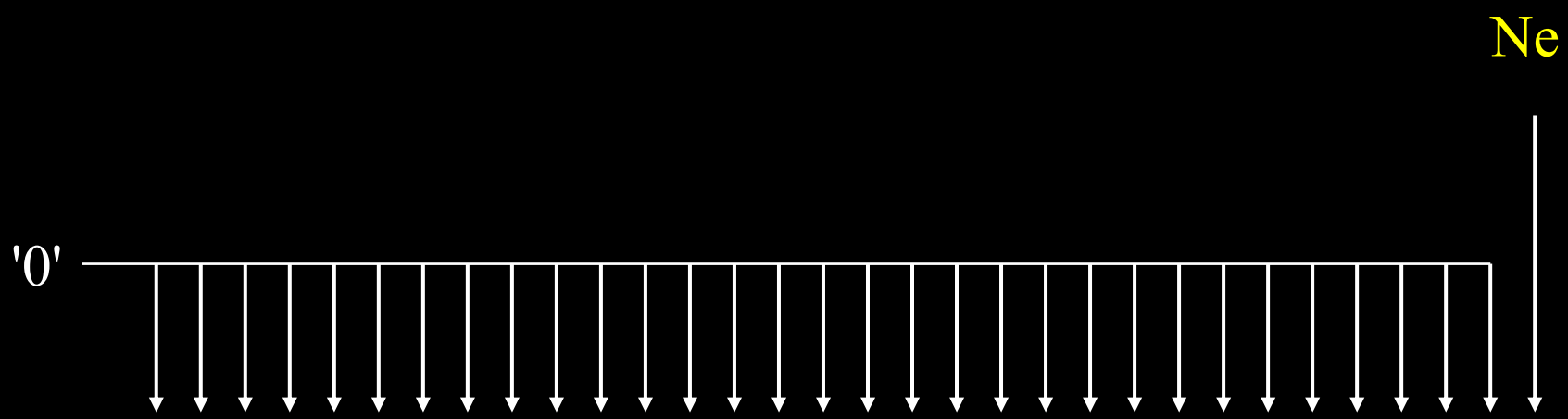
Εμφάνιση του Πρόσημου στην Έξοδο της ALU

- Η υλοποίηση του "**θέσε εάν μικρότερο από (set less than)**" που απαιτούν οι εντολές SLTI και SLT γίνεται ως εξής:
  - Εκτελείται η σύγκριση σαν πράξη αφαίρεσης στον Αφαιρέτη της ALU
    - Εάν το αποτέλεσμα της πράξης είναι αρνητικός αριθμός (δηλαδή ικανοποιείται η συνθήκη της σύγκρισης), τότε το ψηφίο του πρόσημου του αποτελέσματος Ne (το msb) είναι 1
    - Εάν το αποτέλεσμα της πράξης είναι θετικός αριθμός ή μηδέν (δηλαδή δεν ικανοποιείται η συνθήκη της σύγκρισης), τότε το ψηφίο του πρόσημου του αποτελέσματος Ne (το msb) είναι 0
  - Η έξοδος της ALU τίθεται στο 1 ή 0, αντίστοιχα
    - Το ψηφίο του πρόσημου του αποτελέσματος εμφανίζεται σαν λιγότερο σημαντικό ψηφίο της εξόδου της ALU, ενώ τα υπόλοιπα ψηφία είναι όλα μηδέν

      Bus_S[0] = Ne

      Bus_S[i] = 0 (i = 1, .., 31)

Η Μονάδα SLT της ALU



Στην πραγματικότητα δεν υλοποιείται κάποια μονάδα. Απλά οι είσοδοι του τελικού 32-ψήφιου πολυπλέκτη 4 σε 1 της ALU, που σχετίζονται με τη συγκεκριμένη μονάδα, συνδέονται όπως φαίνεται πιο πάνω

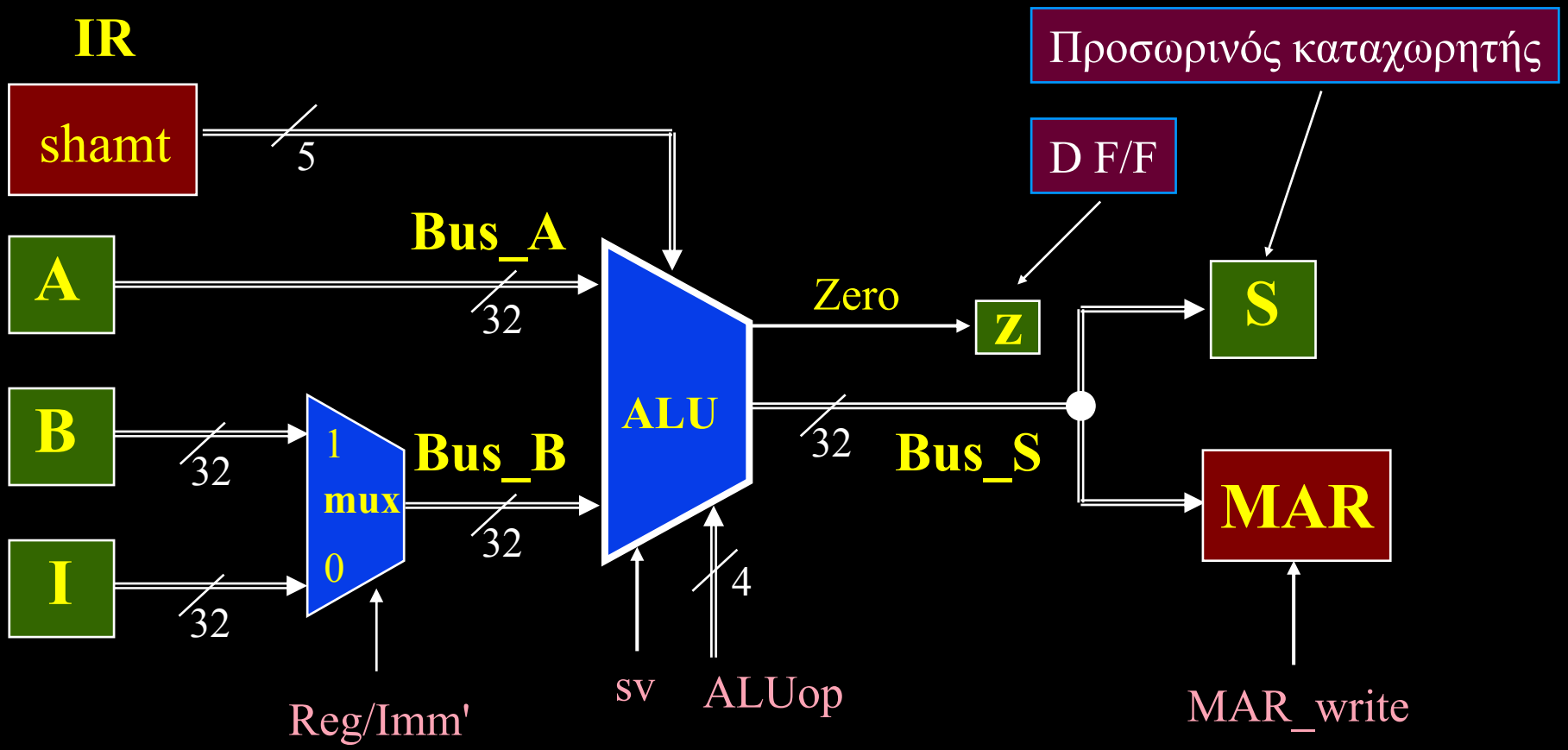
Αριθμητική και Λογική Μονάδα (ALU)

- Ο προσωρινός καταχωρητής **A**, (που περιέχει το περιεχόμενο του καταχωρητή **rs**), συνδέεται με την είσοδο **Bus_A** της ALU
- Ο προσωρινός καταχωρητής **B**, (που περιέχει το περιεχόμενο του καταχωρητή **rt**), ή ο προσωρινός καταχωρητής **I**, (που περιέχει το πεδίο **immediate** του καταχωρητή IR, αφού υποστεί επέκταση πρόσημου ή μηδενός), συνδέεται με την είσοδο **Bus_B** της ALU
  - Η επιλογή γίνεται μέσω ενός **32-ψήφιου πολυπλέκτη 2 σε 1** με σήμα επιλογής το σήμα ελέγχου **Reg/Imm'**
- Ο αριθμός των ολισθήσεων προσδιορίζεται στο πεδίο **shamt** του καταχωρητή IR ή στα 5 λιγότερο σημαντικά ψηφία του προσωρινού καταχωρητή **A**
  - Η επιλογή γίνεται εντός της ALU μέσω ενός **5-ψήφιου πολυπλέκτη 2 σε 1** με σήμα επιλογής το σήμα ελέγχου **sv**

Αριθμητική και Λογική Μονάδα (ALU)

- Στην επόμενη ακμή του ρολογιού η έξοδος Bus_S της ALU αποθηκεύεται πάντα στον προσωρινό καταχωρητή S
- Στην επόμενη ακμή του ρολογιού η έξοδος Bus_S της ALU αποθηκεύεται στον καταχωρητή MAR, μόνο κατά την εκτέλεση των εντολών LW και SW, όταν το σήμα ελέγχου MAR_write = 1
- Στην επόμενη ακμή του ρολογιού το σήμα κατάστασης Zero, που δηλώνει ότι η έξοδος της ALU είναι μηδέν, αποθηκεύεται στο D Flip-Flop Z. Αυτό το σήμα κατάστασης, αν και παράγεται κατά την εκτέλεση όλων των εντολών, χρησιμοποιείται μόνο κατά την εκτέλεση των εντολών BEQ και BNE

Αριθμητική και Λογική Μονάδα (ALU)



Μετά την τοποθέτηση των καταχωρητών στις εισόδους και τις εξόδους της ALU

Μικρο-λειτουργίες της ALU

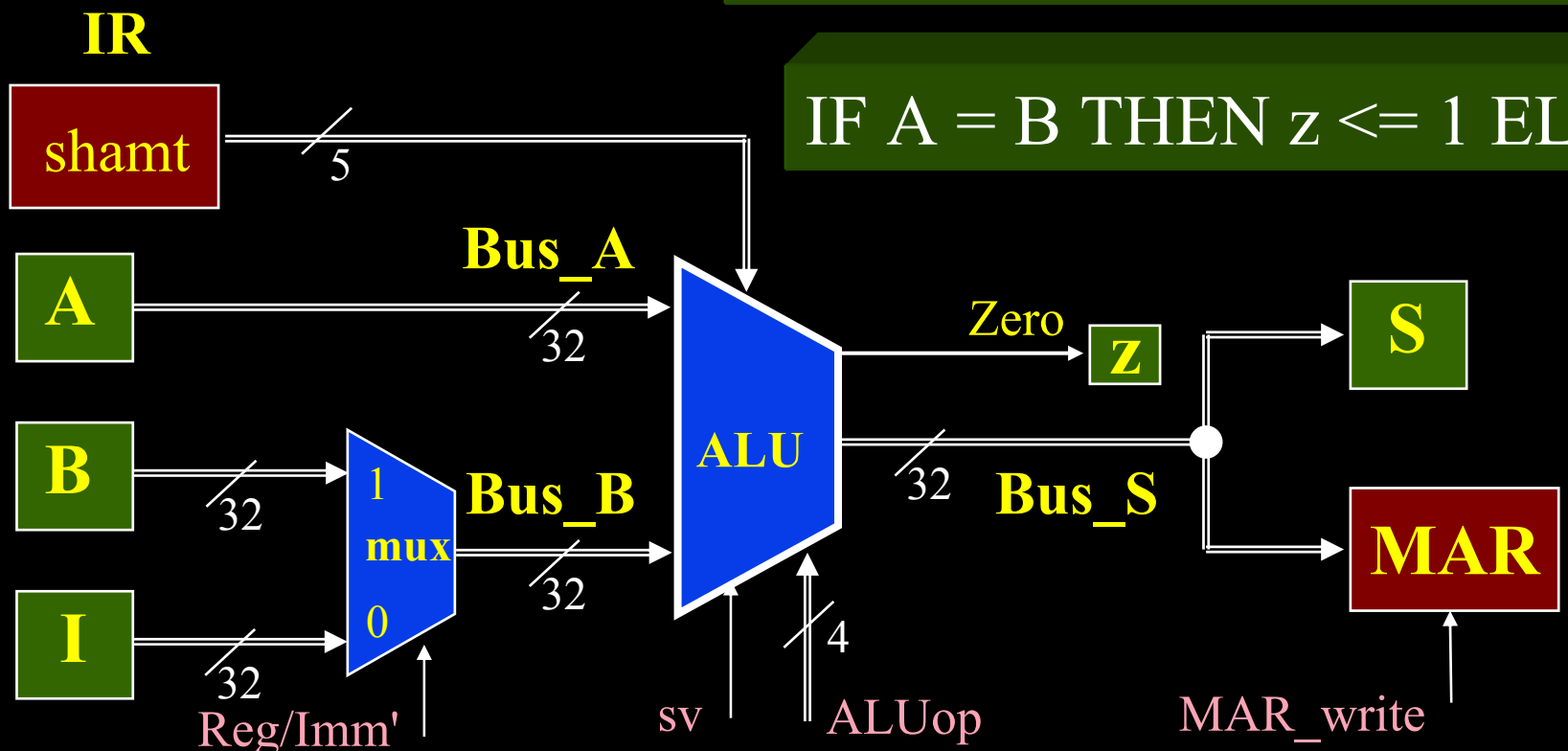
S <= A +/-/and/or/xor/nor I/B

S <= B SLL/SRL/SRA by shamt/A

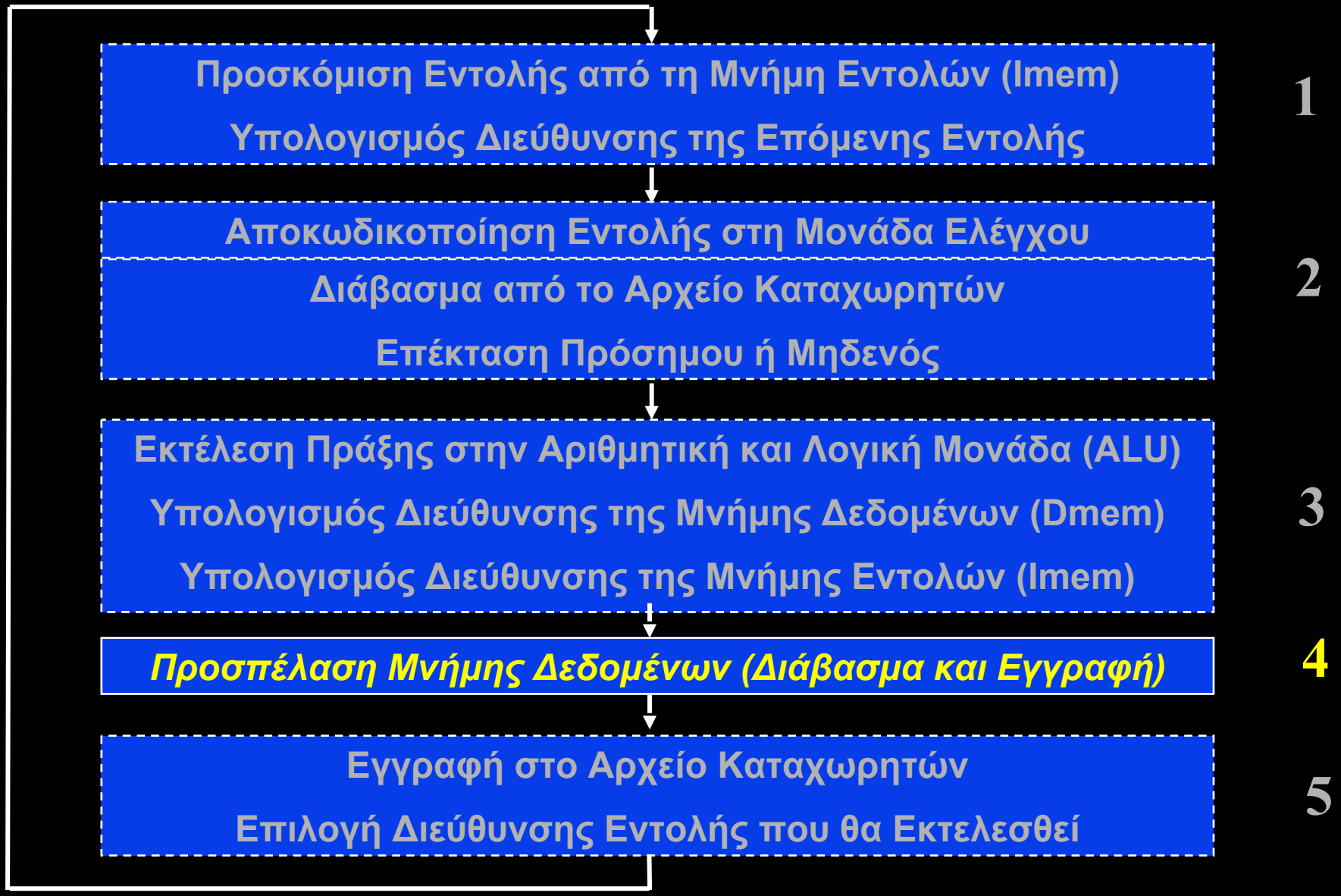
MAR <= A + I

IF A < I/B THEN S <= 1 ELSE S <= 0

IF A = B THEN z <= 1 ELSE z <= 0



Προσπέλαση Μνήμης Δεδομένων Dmem

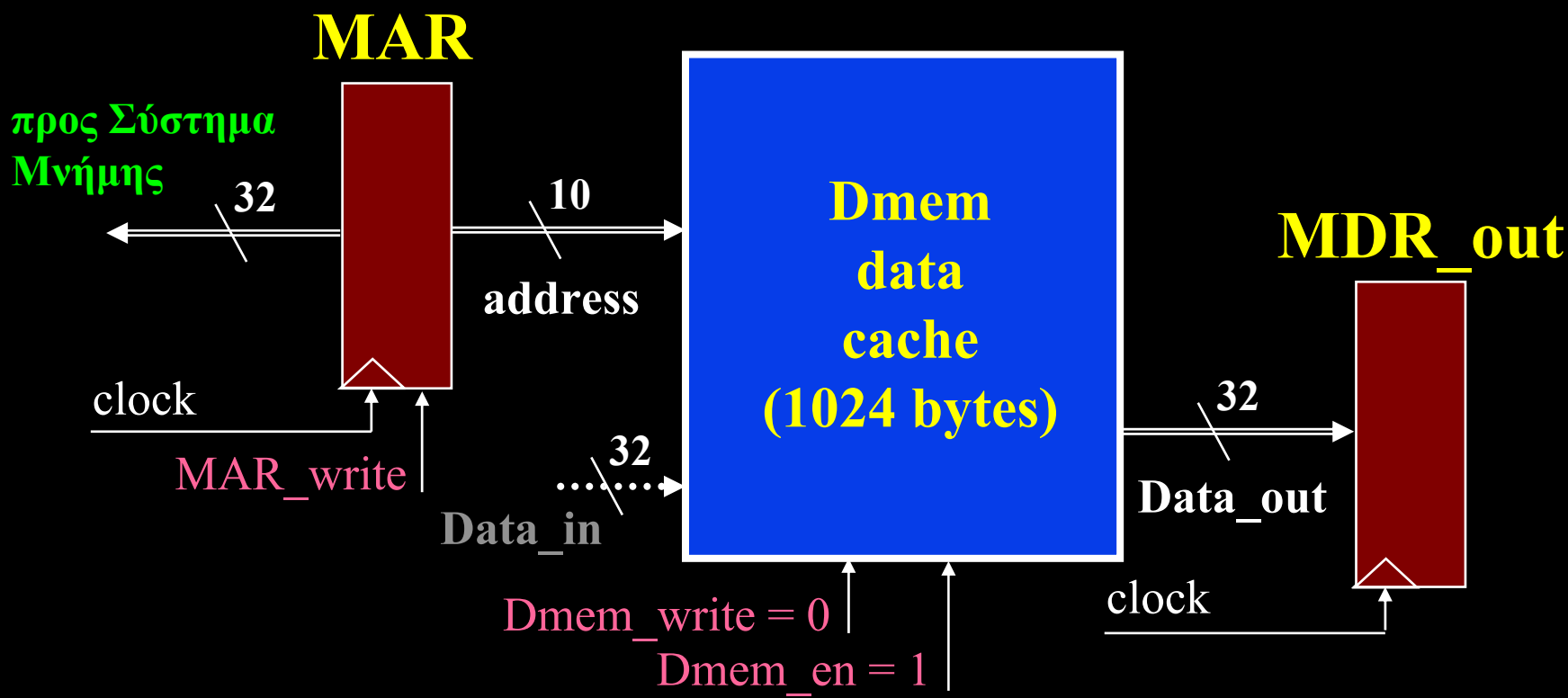


Διάβασμα Μνήμης Δεδομένων Dmem

- Στον καταχωρητή **MAR (Memory Address Register)** είναι αποθηκευμένη η ευθυγραμμισμένη διεύθυνση του περισσότερου σημαντικού byte ενός τελεστέου μεγέθους μίας λέξης (4 bytes) **της μνήμης δεδομένων Dmem** που πρόκειται να διαβασθεί
- Από τη **μνήμη δεδομένων Dmem (Data Memory)** διαβάζεται ο τελεστέος μεγέθους μίας λέξης, όταν **Dmem_en = 1** και **Dmem_write = 0**
- Στον καταχωρητή **MDR_out (Memory Data Register Out)** αποθηκεύεται ο τελεστέος μεγέθους μίας λέξης, που διαβάζεται από τη μνήμη δεδομένων, κατά την επόμενη ακμή του ρολογιού

Διάβασμα Μνήμης Δεδομένων Dmem

MDR_out <= Dmem[MAR]



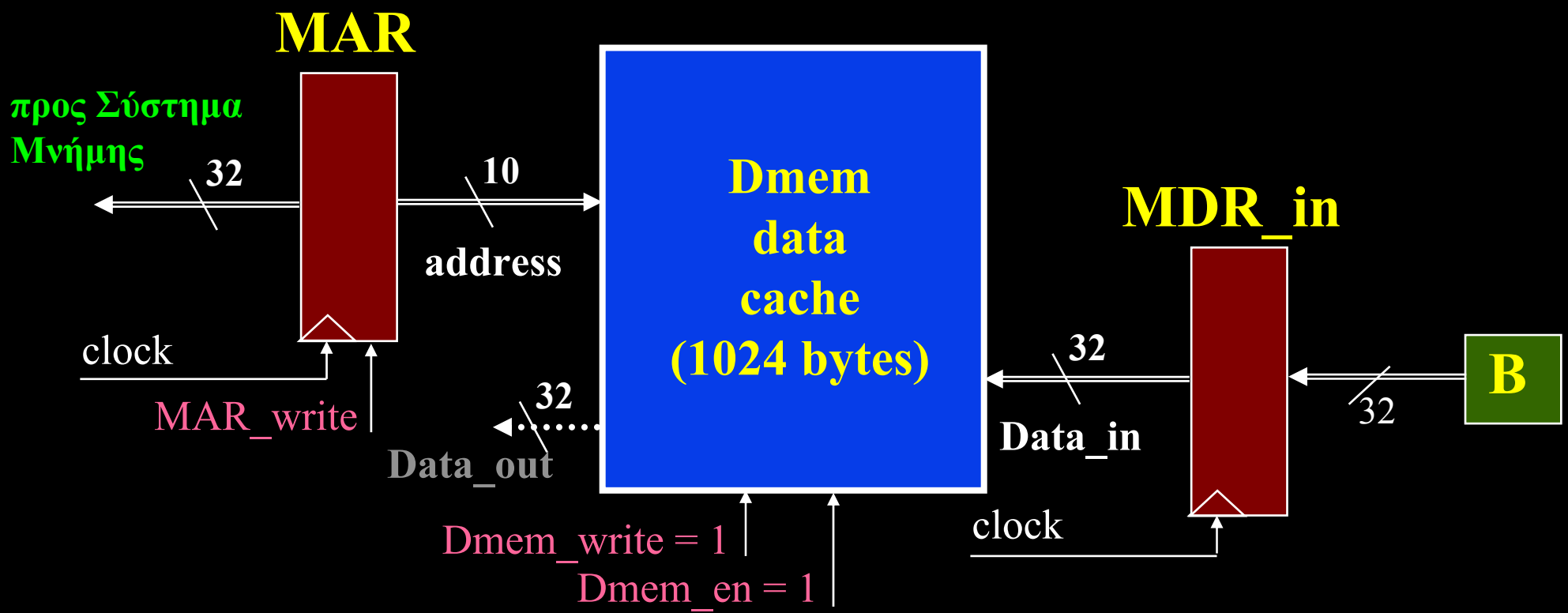
Εγγραφή Μνήμης Δεδομένων Dmem

- Στον καταχωρητή **MAR (Memory Address Register)** είναι αποθηκευμένη η ευθυγραμμισμένη διεύθυνση του περισσότερου σημαντικού byte μίας θέσης μνήμης μεγέθους μίας λέξης (4 bytes) **της μνήμης δεδομένων Dmem** που πρόκειται να εγγραφεί
- Στον καταχωρητή **MDR_in (Memory Data Register In)** είναι αποθηκευμένη η πληροφορία μεγέθους μίας λέξης, που διαβάζεται από τον προσωρινό καταχωρητή **B** και πρόκειται να εγγραφεί στη μνήμη δεδομένων
  - Η μεταφορά της πληροφορίας από τον προσωρινό καταχωρητή **B** στον καταχωρητή **MDR_in** γίνεται στον προηγούμενο κύκλο
- Στη **μνήμη δεδομένων Dmem (Data Memory)** εγγράφεται το περιεχόμενο του καταχωρητή **MDR_in**, όταν **Dmem_en = 1** και **Dmem_write = 1**

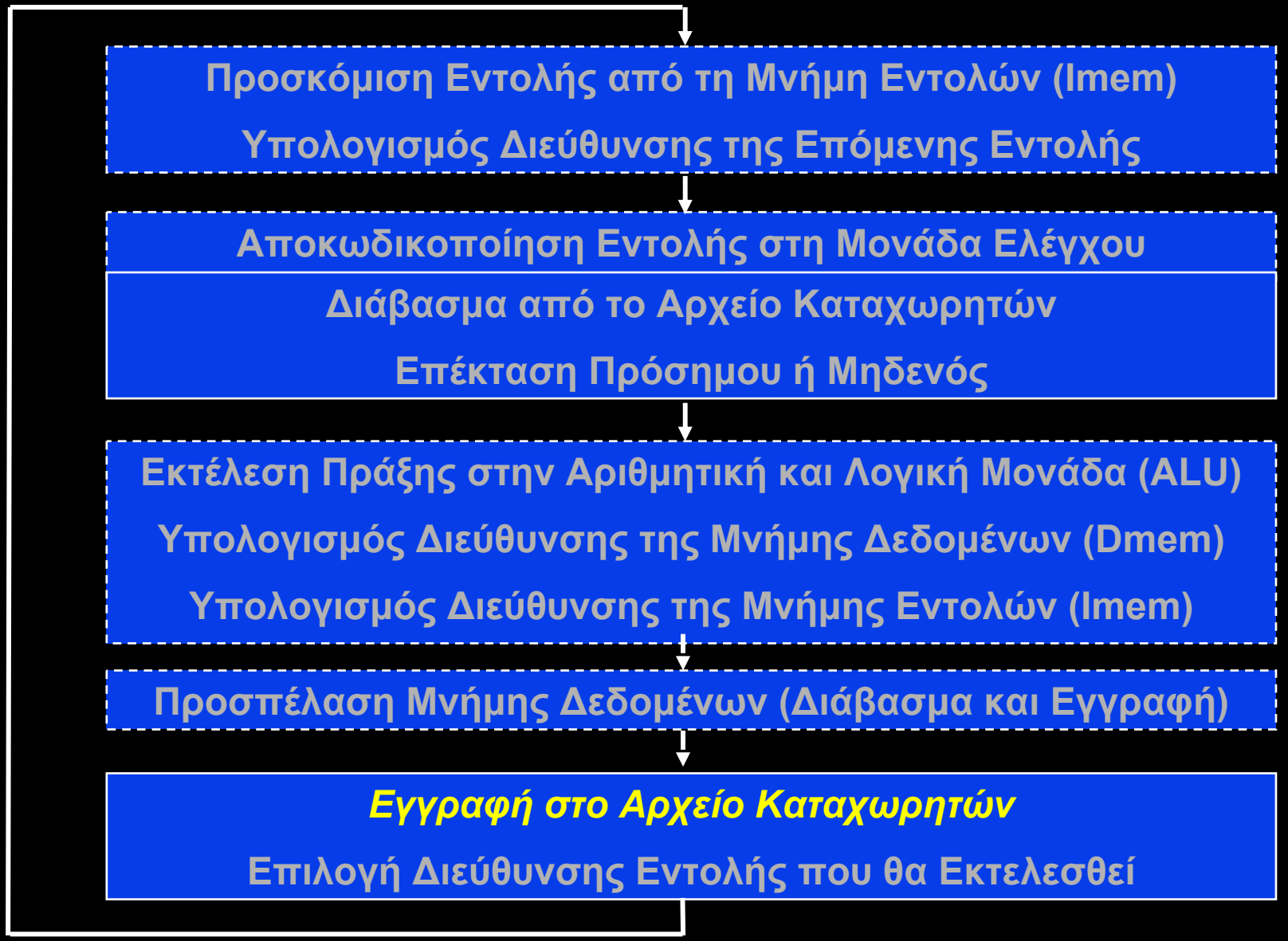
Εγγραφή Μνήμης Δεδομένων Dmem

MDR_in <= B

Dmem[MAR] <= MDR_in



Επιλογή Δεδομένων που Εγγράφονται στο Αρχείο Καταχωρητών

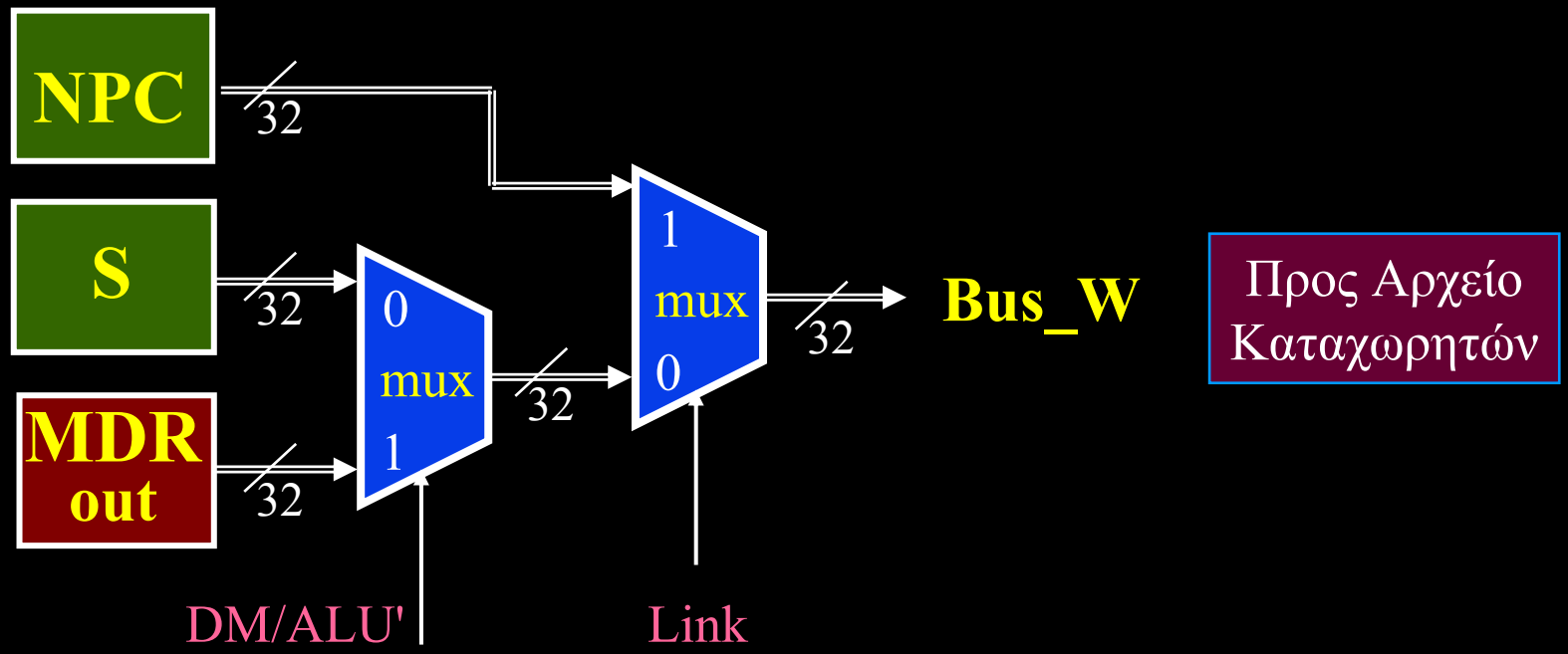


Επιλογή Δεδομένων που Εγγράφονται στο Αρχείο Καταχωρητών

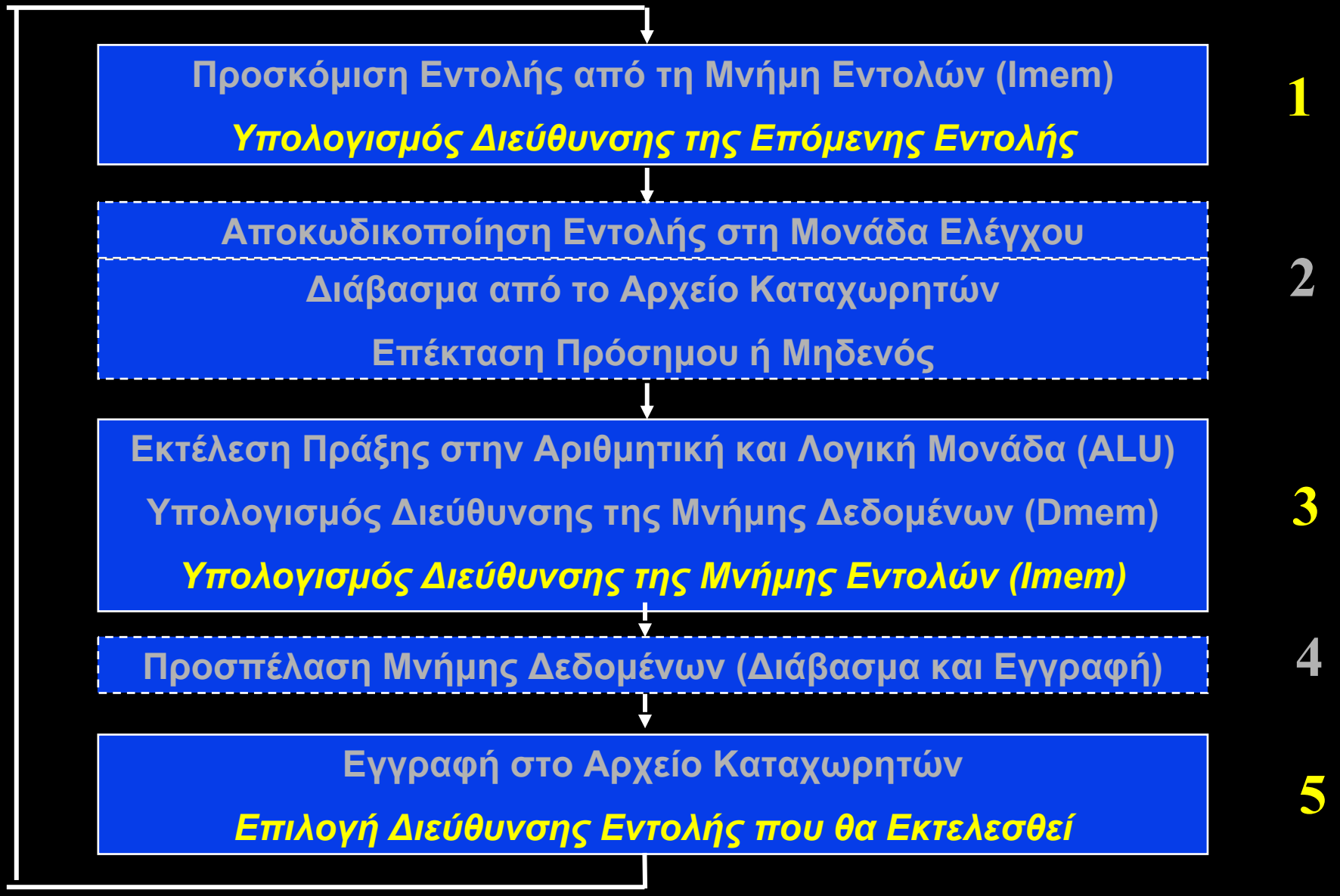
- Για την επιλογή της εγγραφής στο αρχείο καταχωρητών χρησιμοποιούνται δύο σήματα ελέγχου ως εξής :
  - Link : ενεργοποιείται μόνο όταν η εντολή υποστηρίζει τη διαδικασία της σύνδεσης, δηλαδή την αποθήκευση της διεύθυνσης της επόμενης εντολής στο αρχείο καταχωρητών
  - DM/ALU' : επιλέγει σαν πηγή εγγραφής τη μνήμη δεδομένων ή την αριθμητική και λογική μονάδα
- Στο αρχείο καταχωρητών εγγράφονται (μέσω Bus_W) :
  - το αποτέλεσμα πράξης της ALU που αποθηκεύεται προσωρινά στον καταχωρητή S (όταν Link = 0 και DM/ALU' = 0)
  - ο τελεστέος που διαβάζεται από τη μνήμη δεδομένων και αποθηκεύεται προσωρινά στον καταχωρητή MDR_out (όταν Link = 0 και DM/ALU' = 1)
  - η διεύθυνση της επόμενης εντολής (PC+4) που αποθηκεύεται προσωρινά στον καταχωρητή NPC (όταν Link = 1)

Επιλογή Δεδομένων που Εγγράφονται στο Αρχείο Καταχωρητών

Bus_W <= MDR_out/S/NPC



Μονάδα Προσκόμισης Εντολής

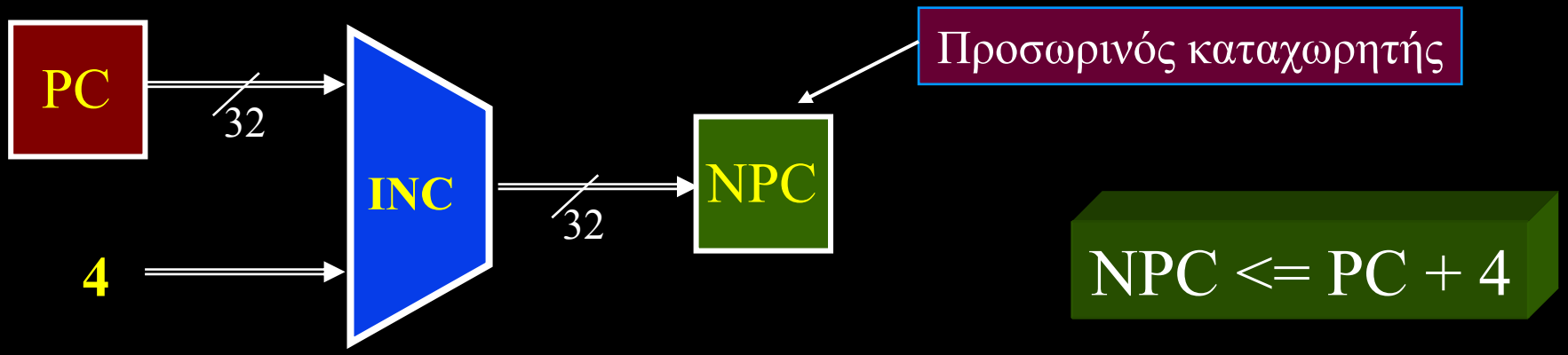


Μονάδα Προσκόμισης Εντολής

- **Η Μονάδα Προσκόμισης Εντολής (Instruction Fetch Unit)**
  - Χρησιμοποιείται για τον **υπολογισμό** και την **επιλογή** της διεύθυνσης της εντολής που πρόκειται να εκτελεσθεί
  - Υπολογίζει τη νέα τιμή του PC ως εξής :
    - **PC <= PC + 4**
      (για ακολουθιακή εκτέλεση όλων των εντολών
      (εκτός των BEQ, BNE, JR και JALR)
    - **PC <= PC + 4**
      (όταν δεν ικανοποιείται η συνθήκη στις εντολές BEQ και BNE)
    - **PC <= PC + 4 + 4 x sign_extend(immediate)**
      (όταν ικανοποιείται η συνθήκη στις εντολές BEQ και BNE)
    - **PC <= reg[rs]**
      (στις εντολές JR και JALR)

Υπολογισμός της Διεύθυνσης PC + 4

- Είναι κοινός για όλες τις εντολές
- Χρησιμοποιείται ένας απλοποιημένος αθροιστής των 32 ψηφίων, του οποίου η μία είσοδος είναι πάντα 4 (που ονομάζεται αυξητής κατά 4 – incrementer by 4)
- Η αύξηση κατά 4 προκύπτει ως εξής:
  - Η μνήμη εντολών διευθυνσιοδοτείται ανά byte
  - Κάθε εντολή έχει μέγεθος 4 bytes
- Στην επόμενη ακμή του ρολογιού η διεύθυνση της επόμενης εντολής (PC+4) αποθηκεύεται στον προσωρινό καταχωρητή NPC

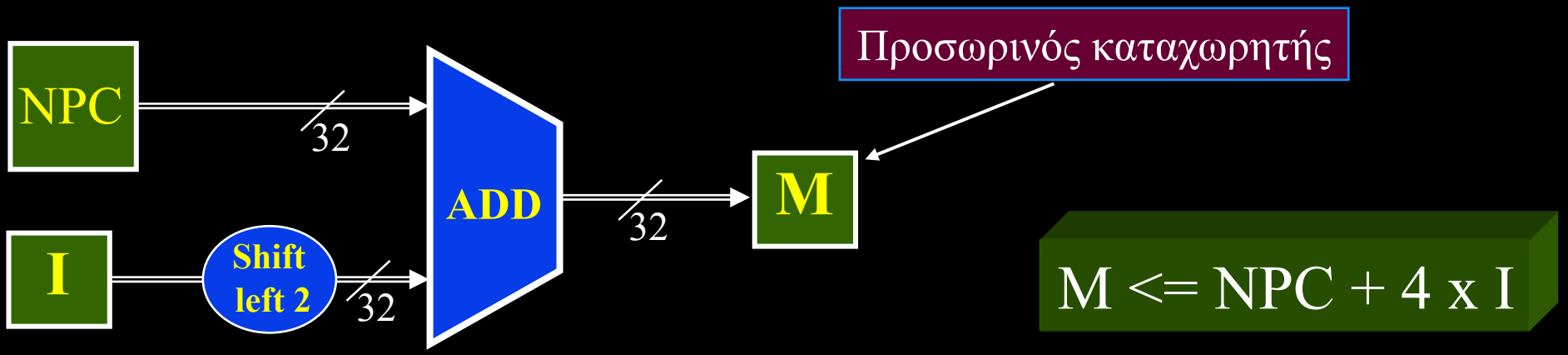


Υπολογισμός της Διεύθυνσης PC+4+4μ

- Απαιτείται στις εντολές διακλάδωσης με συνθήκη που υλοποιούνται με PC-σχετική διευθυνσιοδότηση (BEQ, BNE)
- Στο πεδίο immediate της εντολής έχει αποθηκευθεί σαν μετατόπιση η διαφορά σε εντολές μ της εντολής, που θα εκτελεσθεί όταν ικανοποιείται η συνθήκη, από την εντολή που έπεται της εντολής διακλάδωσης.
- Στη δίοδο δεδομένων του επεξεργαστή υπολογίζεται η διαφορά σε bytes 4μ μεγέθους 32 ψηφίων ως εξής:
  - Η διαφορά σε εντολές μ, που είναι αποθηκευμένη στο πεδίο immediate του καταχωρητή IR, επεκτείνεται στα 32 ψηφία με επέκταση πρόσημου και αποθηκεύεται στον προσωρινό καταχωρητή I
  - Η διαφορά σε εντολές (32 ψηφίων) που είναι αποθηκευμένη στον προσωρινό καταχωρητή I μετατρέπεται σε διαφορά σε bytes 4μ (32 ψηφίων) με αριστερή ολίσθηση κατά 2 ψηφία (πολλαπλασιασμός επί 4)

Υπολογισμός της Διεύθυνσης PC+4+4μ

- Η **διαφορά σε bytes 4μ** προστίθεται στη **διεύθυνση της επόμενης εντολής (PC+4)**, ώστε να προκύψει η **διεύθυνση προορισμού διακλάδωσης (PC+4+4μ)** με τη χρήση ενός χωριστού αθροιστή των 32 ψηφίων
- Στην επόμενη ακμή του ρολογιού η **διεύθυνση προορισμού διακλάδωσης (PC+4+4μ)** αποθηκεύεται στον προσωρινό καταχωρητή **M**



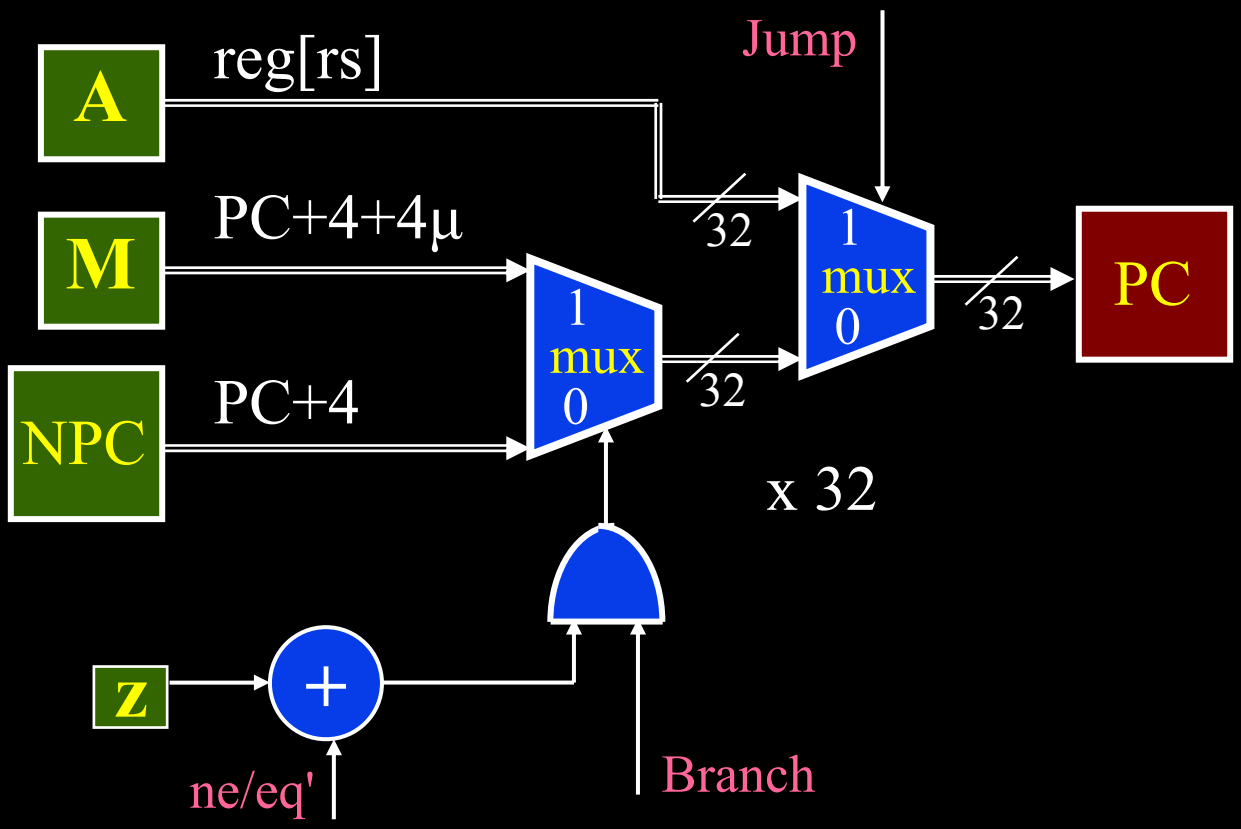
Επιλογή Διεύθυνσης Επόμενης Εντολής

- Για την επιλογή της διεύθυνσης της επόμενης εντολής χρησιμοποιούνται τρία σήματα ελέγχου ως εξής:
  - Branch : ενεργοποιείται όταν εκτελείται εντολή διακλάδωσης
  - Jump : ενεργοποιείται όταν εκτελείται εντολή μεταπήδησης
  - ne/eq' : επιλέγει τη συνθήκη στις εντολές διακλάδωσης BNE/BEQ
- Εάν Jump = 1, τότε PC = A
  (για τις εντολές JR και JALR)
- Εάν Branch = 0 και Jump = 0, τότε PC = NPC
  (για όλες τις εντολές, εκτός BEQ, BNE, JR και JALR)
- Εάν Branch = 1, ne/eq' = 0 και Jump = 0,
  τότε PC = M (όταν z = 1) ή PC = NPC (όταν z = 0)
  (για την εντολή BEQ)
- Εάν Branch = 1, ne/eq' = 1 και Jump = 0,
  τότε PC = M (όταν z = 0) ή PC = NPC (όταν z = 1)
  (για την εντολή BNE)

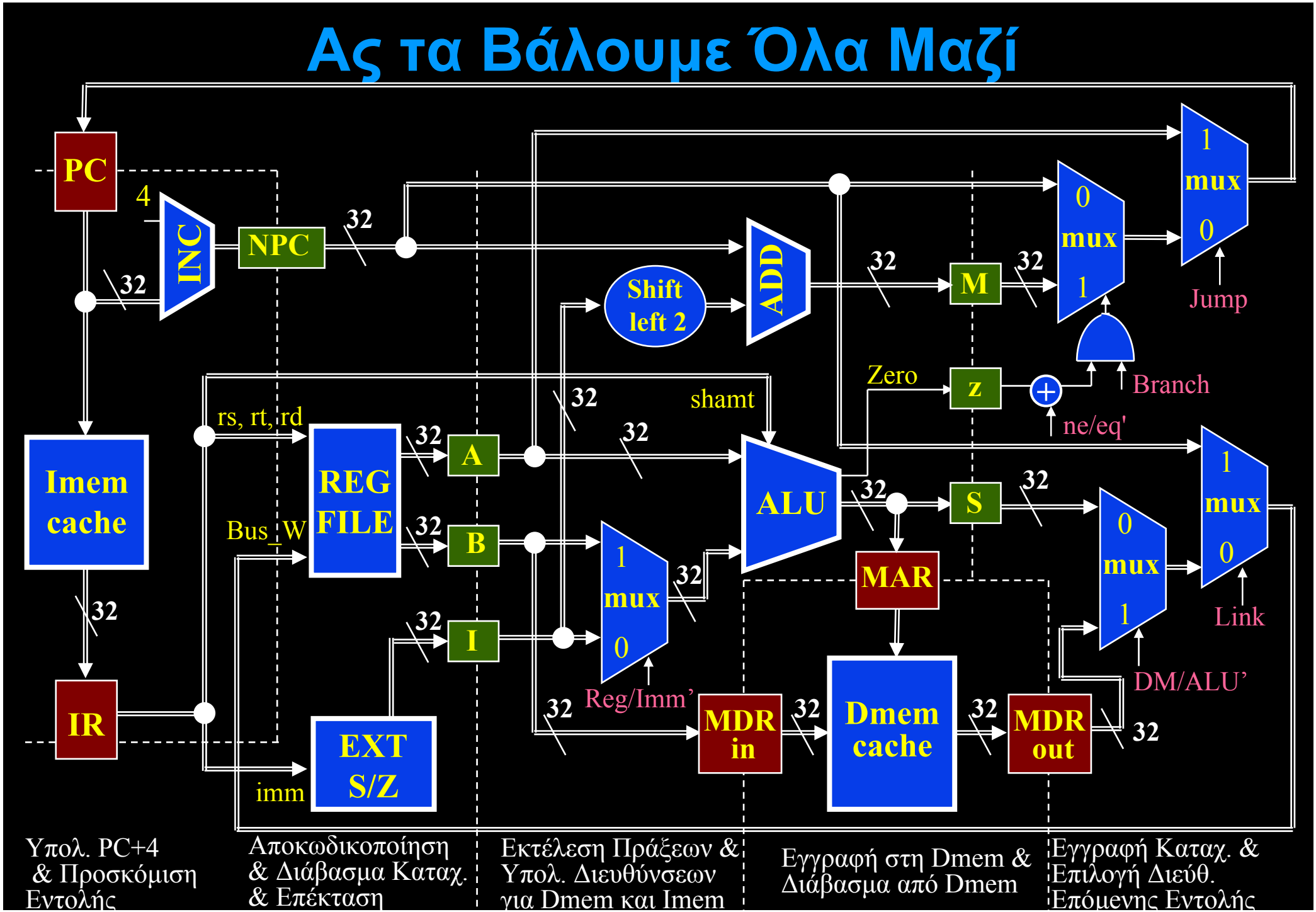
Επιλογή Διεύθυνσης Επόμενης Εντολής

PC <= NPC/A

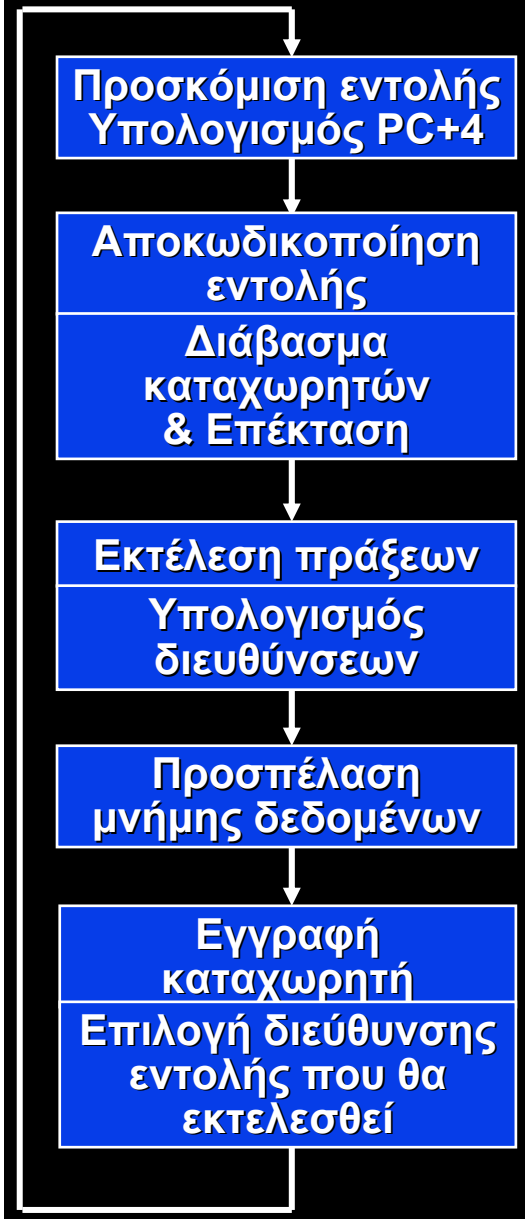
IF z = 1/0 THEN PC <= M ELSE PC <= NPC



Ας τα Βάλουμε Όλα Μαζί



Μικρο-Λειτουργίες και Σήματα Ελέγχου




```
IR <= Imem[PC]
NPC <= PC + 4
```

```
A <= reg[rs]
B <= reg[rt]
I <=sign/zero_extend[imm]
```

```
MAR <= A + I
S <= A +/-/and/or/xor/nor I/B
S <= B SLL/SRL/SRA by shamt/A
If A < I/B then S <= 1 else S <= 0
If A = B then z <= 1 else z <= 0
MDR_in <= B
M <= NPC + 4 x I
```

```
MDR_out <= Dmem[MAR]
Dmem[MAR] <= MDR_in
```

```
reg[rt/rd] <= S/MDR_out/NPC
PC <= NPC/A
If z = 1/0 then PC <= M
           else  PC <= NPC
```

Imem_write = 0 IR_write = 1	1
Sign/Zero’ = 1/0/x	2
Reg/Imm’ = 1/0/x ALUop[3:0] = Βλέπε στον επόμενο πίνακα sv = 1/0/x MAR_write = 1/0	3
Dmem_write = 1/0	4
Reg_write = 1/0 Reg/Imm' = 1/0/x DM/ALU’ = 1/0/x Link = 1/0/x Jump = 1/0 Branch = 1/0 ne/eq' = 1/0/x PC_write = 1	5

Imem_en = Dmem_en = 1 πάντα

Επεξηγήσεις στα Σήματα Ελέγχου

- Κάθε **φάση** εκτελείται σε έναν **κύκλο ρολογιού**
- Σε κάθε κύκλο αναφέρονται μόνο εκείνα τα σήματα ελέγχου, που απαιτούνται για να ελεγχθούν οι μικρο-λειτουργίες του συγκεκριμένου κύκλου
  - στους κύκλους που δεν αναφέρονται η τιμή τους είναι αδιάφορη
  - κάποια σήματα ελέγχου ελέγχουν μικρο-λειτουργίες σε περισσότερους από έναν κύκλο
- Τα σήματα ελέγχου ενεργοποίησης εγγραφής (X_write) εμφανίζονται σε συγκεκριμένο κύκλο που απαιτείται αντίστοιχος έλεγχος μνήμης ή καταχωρητή
  - στον κύκλο αυτόν έχουν την τιμή 1, όταν πρόκειται για εγγραφή είτε στη μνήμη εντολών ή δεδομένων, είτε σε καταχωρητή
  - στον κύκλο αυτόν έχουν την τιμή 0, όταν πρόκειται για διάβασμα από τη μνήμη εντολών ή δεδομένων
  - στους υπόλοιπους κύκλους, όπου δεν αναφέρονται, έχουν πάντα την τιμή 0

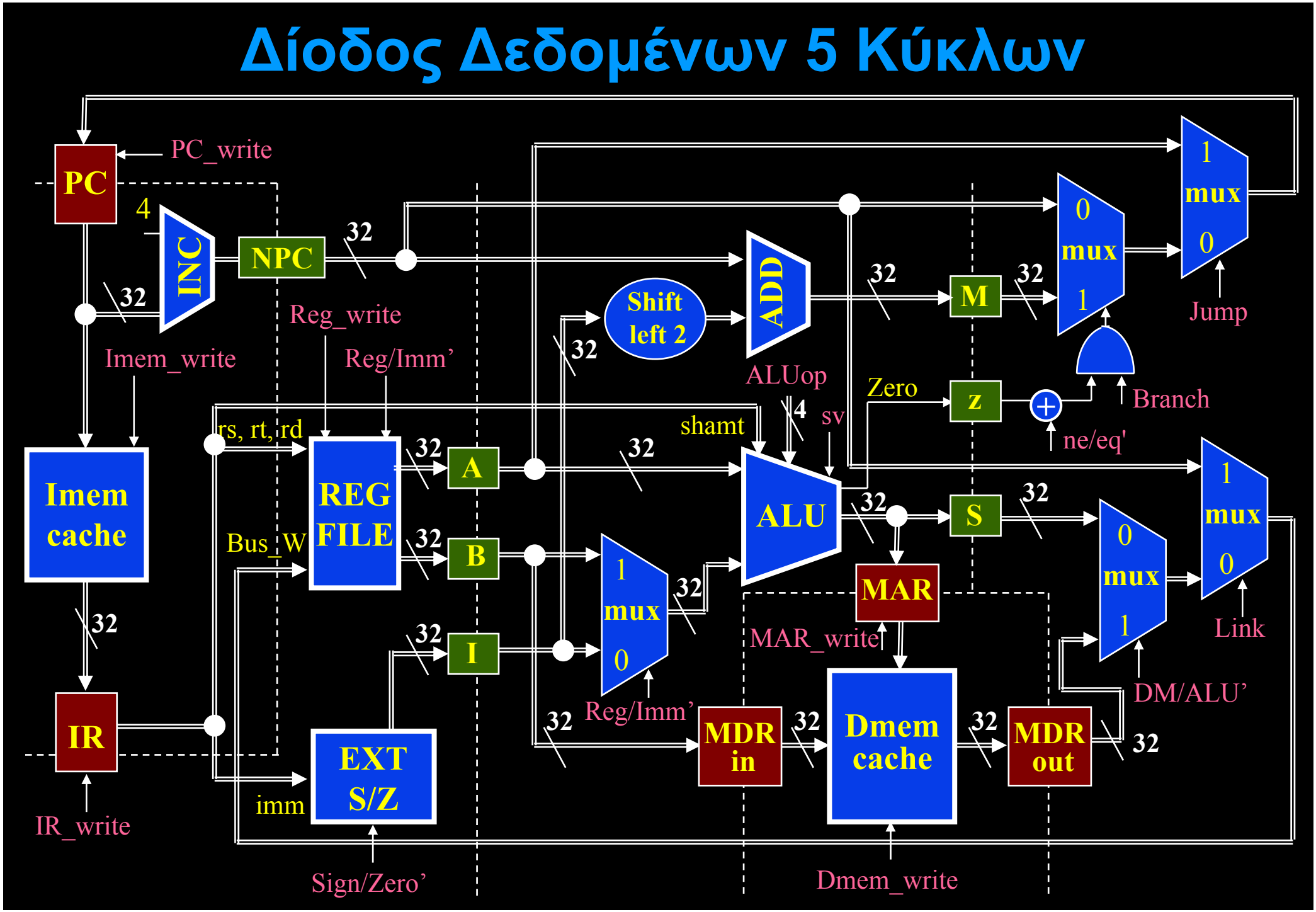
Επίδραση των Σημάτων Ελέγχου

		τιμή = 0	τιμή = 1
1	Imem_write Imem_en IR_write	Διάβασμα από Imem Απενεργοποίηση Imem Τίποτα	Εγγραφή στην Imem Ενεργοποίηση Imem Εγγραφή στον IR
2	Sign/Zero’	Επέκταση μηδενός	Επέκταση πρόσημου
3	Reg/Imm’ sv MAR_write	Bus_B (ALU) = I σταθερός αριθμός ολισθ. Τίποτα	Bus_B (ALU) = B μεταβλητός αριθμός ολ. Εγγραφή στον MAR
4	Dmem_write Dmem_en	Διάβασμα από Dmem Απενεργοποίηση Dmem	Εγγραφή στην Dmem Ενεργοποίηση Dmem
5	Reg_write Reg/Imm' DM/ALU’ (Link=0) Link =1 Jump Branch ne/eq' PC_write	Τίποτα Εγγραφή στον rt Bus_W = S Bus_W = S/MDR_out Υπόλοιπες εντολές Υπόλοιπες εντολές Διακλάδωση, όταν ίσο Τίποτα	Εγγραφή στο Reg. File Εγγραφή στον rd Bus_W = MDR_out Bus_W = NPC Εντολές JR και JALR Εντολές BEQ και BNE Διακλάδωση, όταν άνισο Εγγραφή στον PC

Το Σήμα Ελέγχου ALUop[3:0] της ALU

ALUop	Εντολές	Λειτουργία ALU
10 01	ADDU, LW, SW	Προσημασμένη Πρόσθεση Χωρίς Υπερχείλιση
10 11	SUBU	Προσημασμένη Αφαίρεση Χωρίς Υπερχείλιση
10 11	BEQ, BNE	Προσημασμένη Αφαίρεση & Εμφάνιση Μηδενός
11 00	AND	Λογικό AND
11 01	OR	Λογικό OR
11 10	XOR	Λογικό XOR
11 11	NOR	Λογικό NOR
00 00	SLL, SLLV	Αριστερή Λογική Ολίσθηση
00 10	SRL, SRLV	Δεξιά Λογική Ολίσθηση
00 11	SRA, SRAV	Δεξιά Αριθμητική Ολίσθηση
01 10	SLTI, SLT	Προσημασμένη Αφαίρεση & Εμφάνιση Πρόσημου

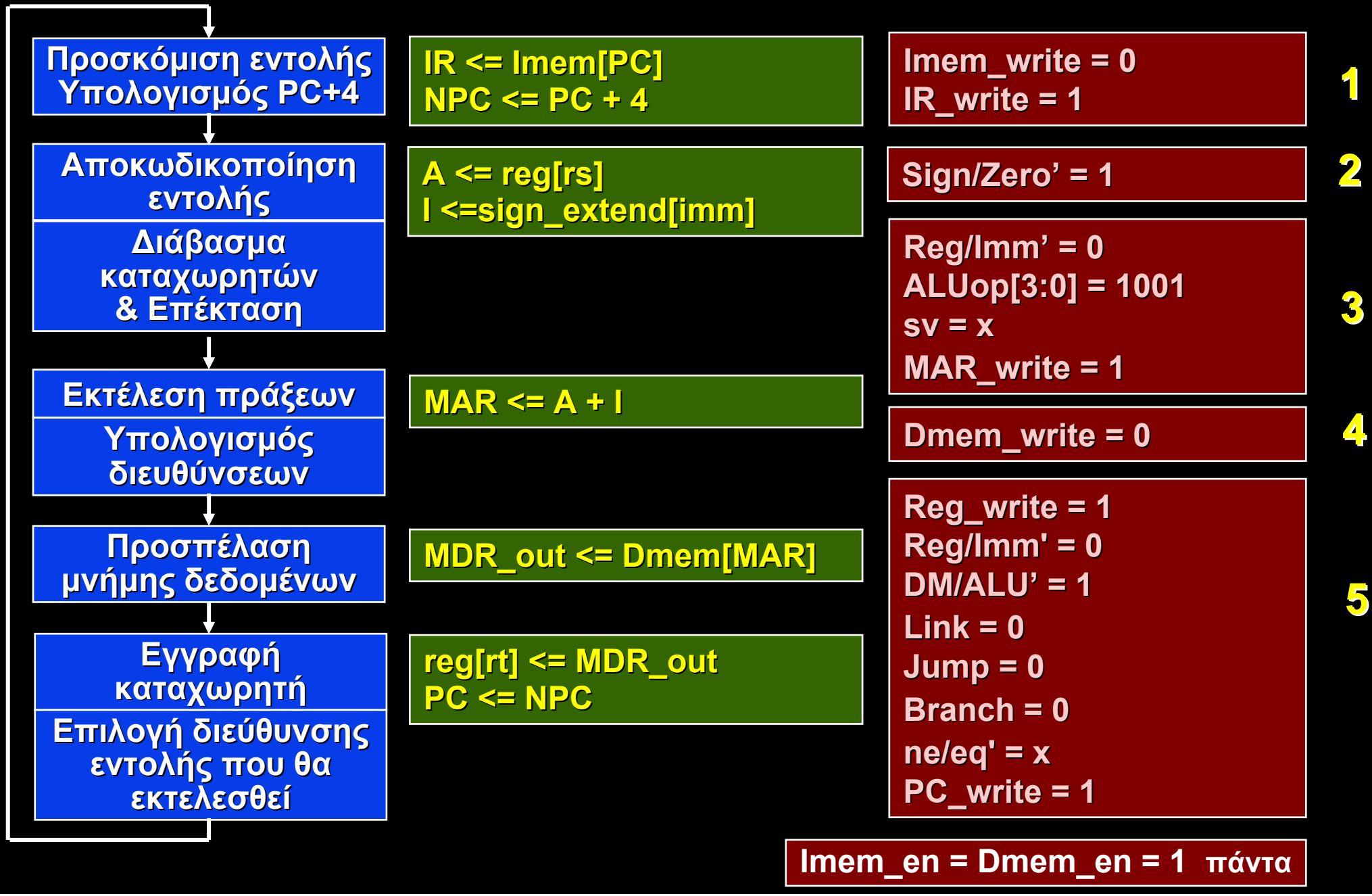
Δίοδος Δεδομένων 5 Κύκλων



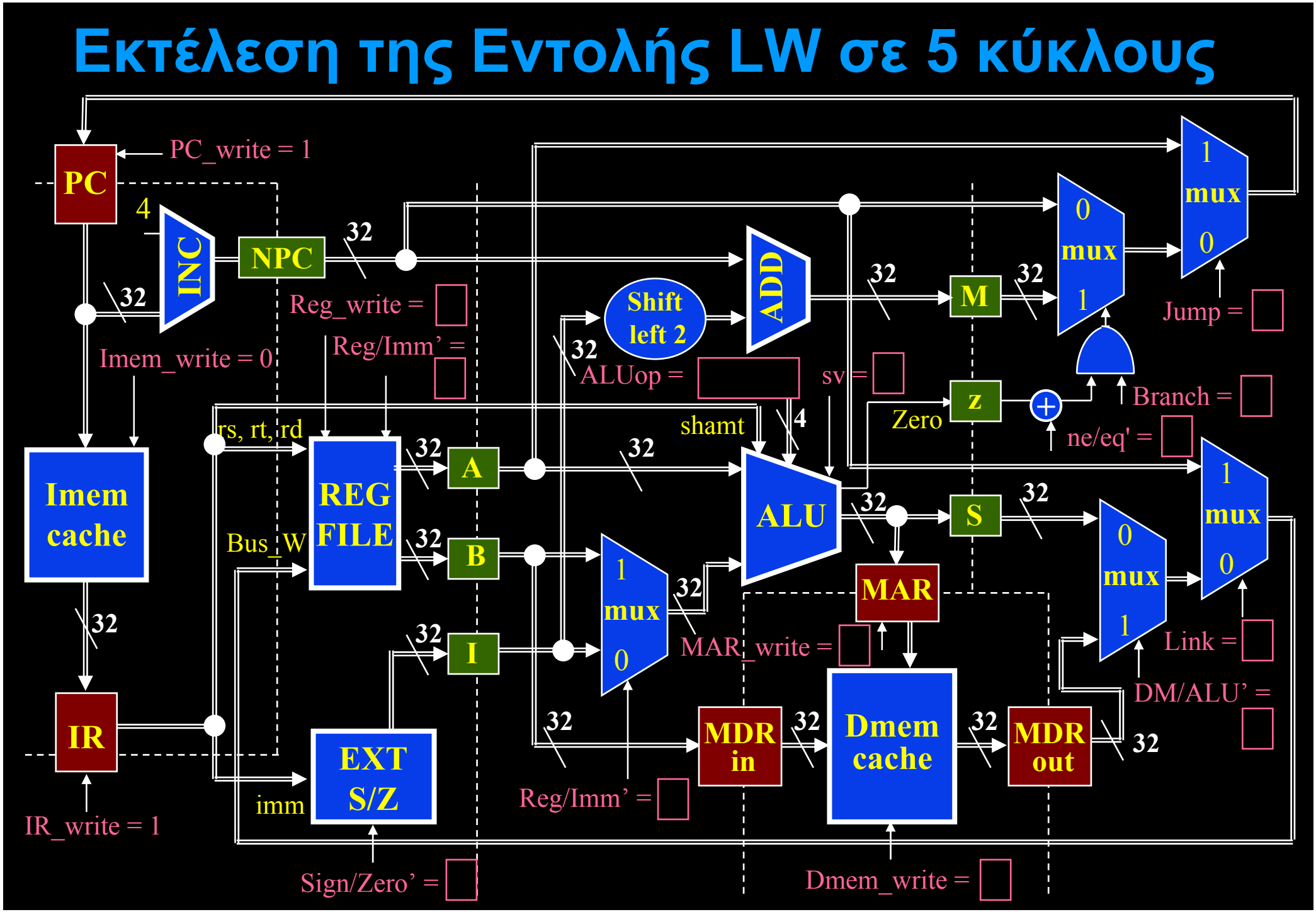
Εκτέλεση της Εντολής LW σε 5 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Επέκταση πρόσημου του πεδίου **immediate**
- **3ος κύκλος**
  - Εκτέλεση της πρόσθεσης στην ALU
    (για τον υπολογισμό της διεύθυνσης της Μνήμης Δεδομένων)
- **4ος κύκλος**
  - Διάβασμα από τη Μνήμη Δεδομένων
- **5ος κύκλος**
  - Εγγραφή στον καταχωρητή **rt** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

Μικρο-λειτουργίες και Σήματα Ελέγχου της LW



Εκτέλεση της Εντολής LW σε 5 κύκλους

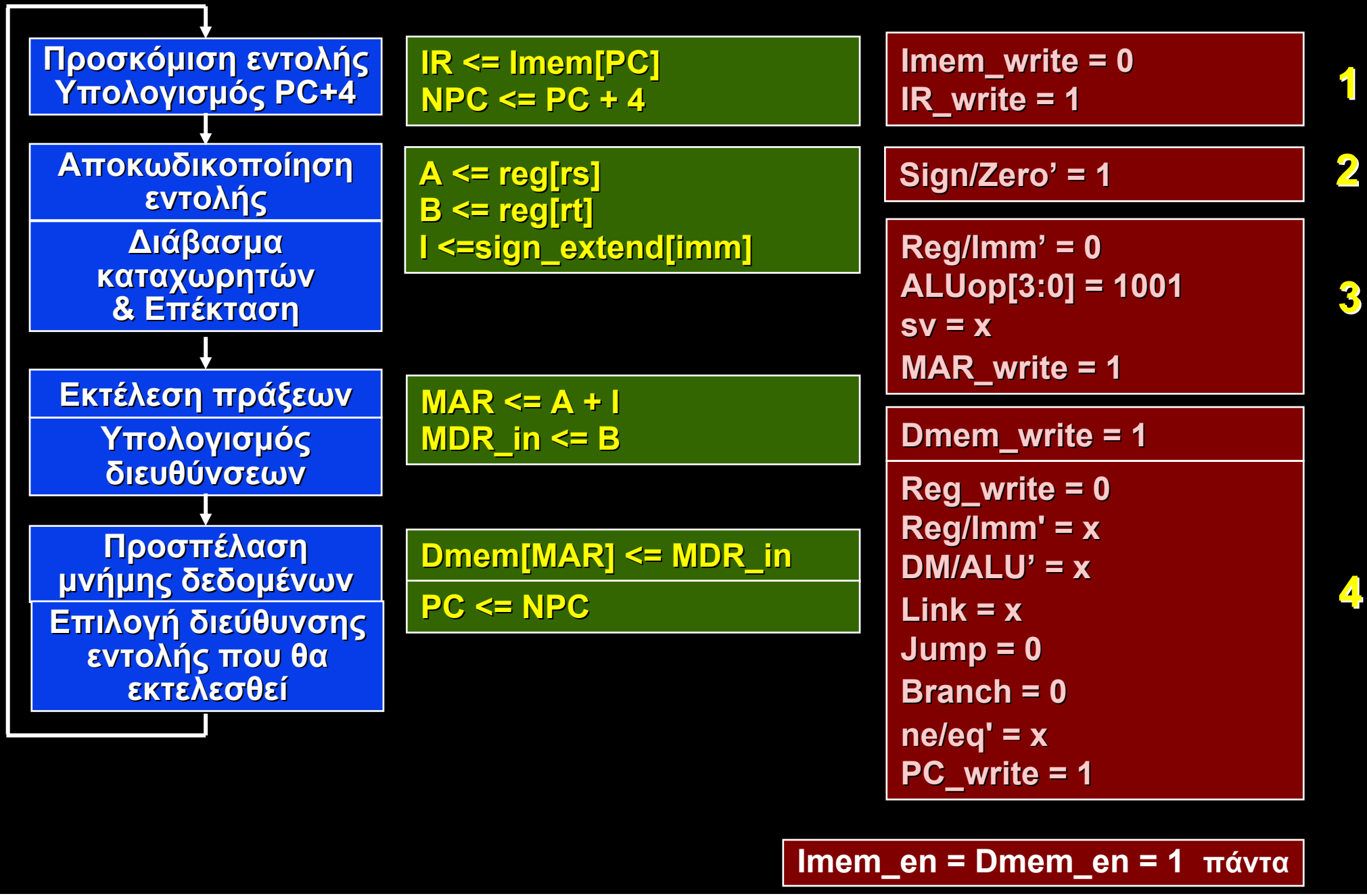


Εκτέλεση της Εντολής SW σε 4 κύκλους

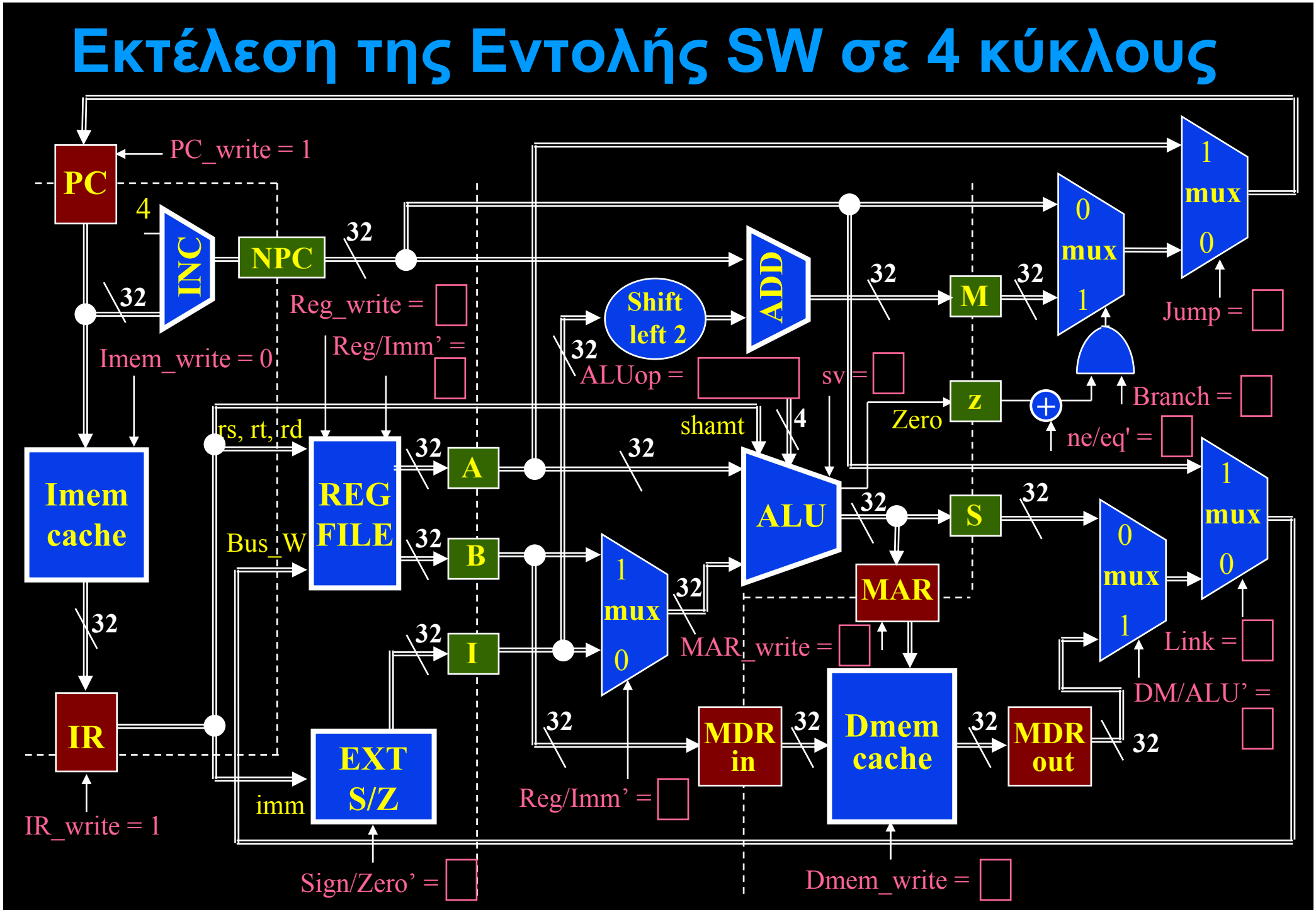
- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Διάβασμα του καταχωρητή **rt** από το Αρχείο Καταχωρητών
  - Επέκταση πρόσημου του πεδίου **immediate**
- **3ος κύκλος**
  - Εκτέλεση της πρόσθεσης στην ALU
    (για τον υπολογισμό της διεύθυνσης της Μνήμης Δεδομένων)
- **4ος κύκλος**
  - Εγγραφή στη Μνήμη Δεδομένων
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

Η τέταρτη και η πέμπτη φάση εκτελούνται παράλληλα στον τέταρτο κύκλο

Μικρο-λειτουργίες και Σήματα Ελέγχου της SW



Εκτέλεση της Εντολής SW σε 4 κύκλους

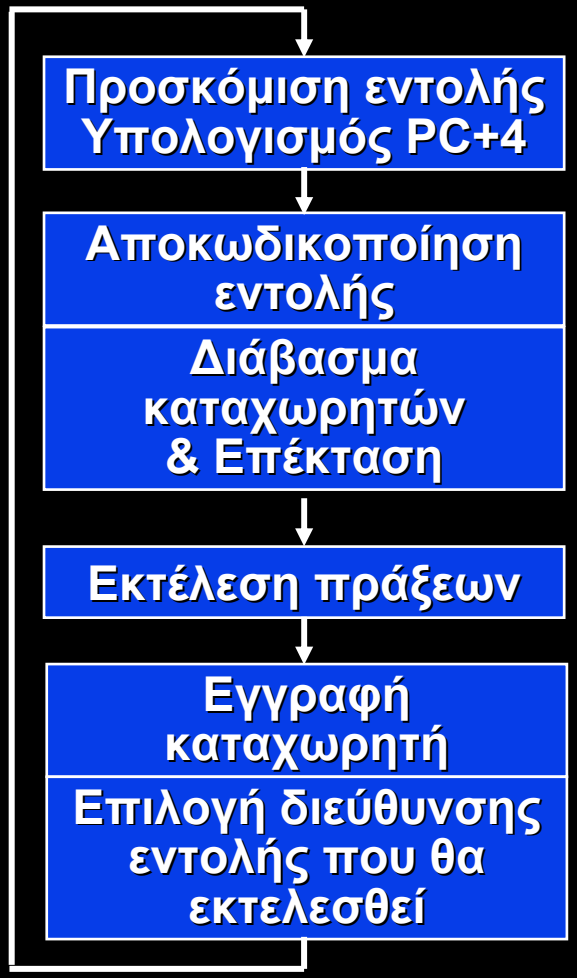


Εκτέλεση της Εντολής ADDIU* σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Επέκταση πρόσημου (ADDI) ή μηδενός (ANDI, ORI, XORI) του πεδίου **immediate**
- **3ος κύκλος**
  - Εκτέλεση της πράξης (+/and/or/xor) στην ALU
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή **rt** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

* Παρόμοια οι εντολές ANDI, ORI, XORI

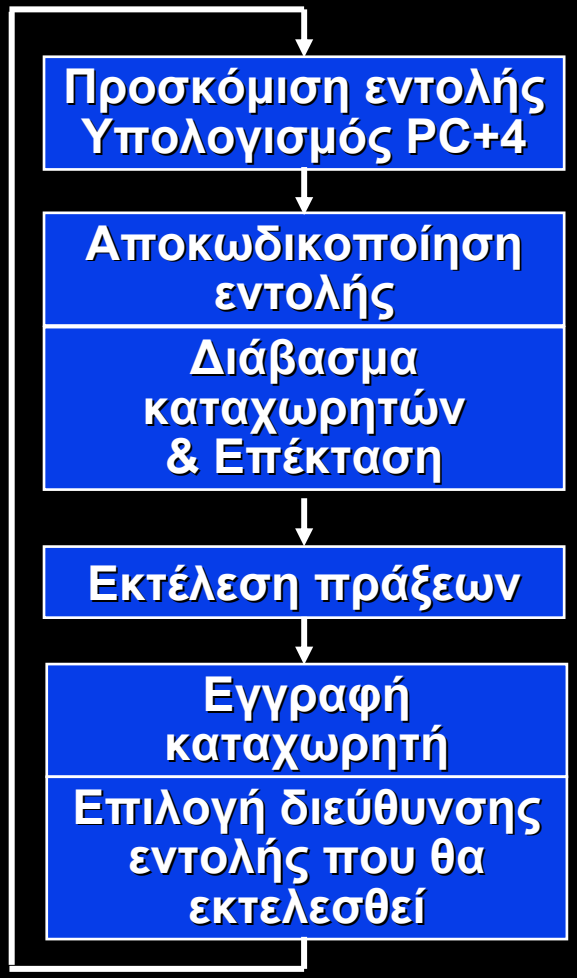
Μικρο-λειτουργίες και Σήματα Ελέγχου της ADDIU



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] I <=sign_extend[imm]	Sign/Zero’ = 1	2
	Reg/Imm’ = 0 ALUop[3:0] = 1001 sv = x MAR_write = 0	3
S <= A + I	Dmem_write = 0	
reg[rt] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 0 DM/ALU’ = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

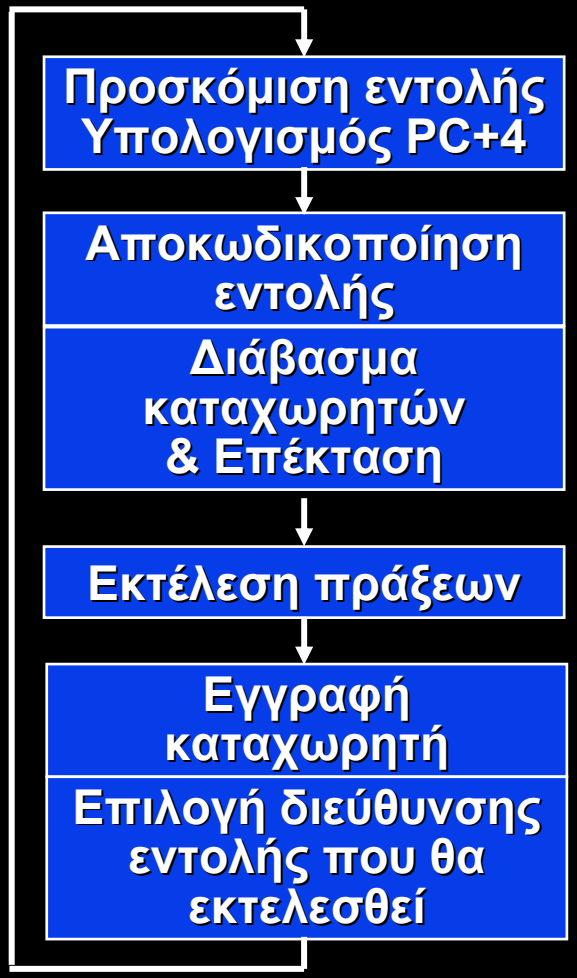
Μικρο-λειτουργίες και Σήματα Ελέγχου της ANDI



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] I <= zero_extend[imm]	Sign/Zero' = 0	2
	Reg/Imm' = 0 ALUop[3:0] = 1100 sv = x MAR_write = 0	3
S <= A and I	Dmem_write = 0	
reg[rt] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 0 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

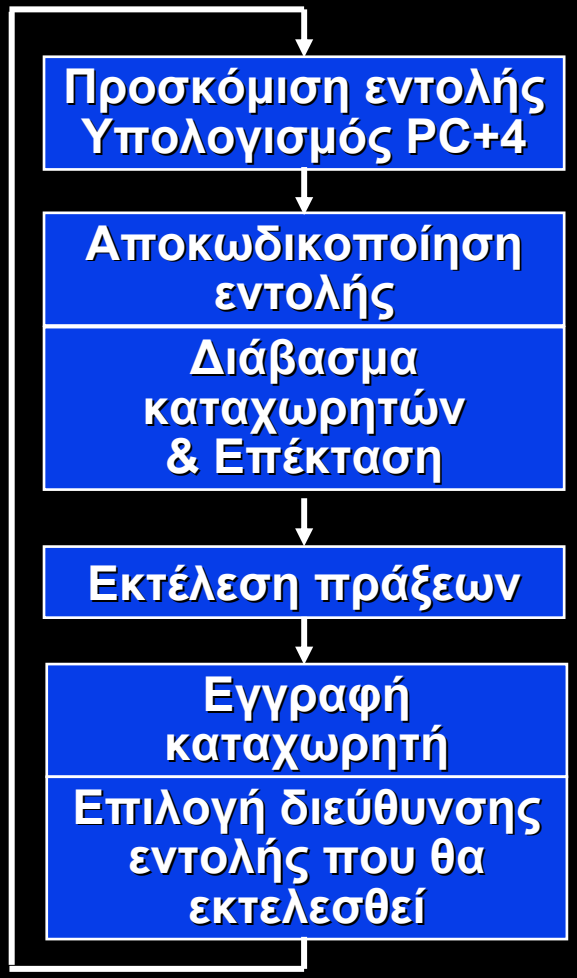
Μικρο-λειτουργίες και Σήματα Ελέγχου της ORI



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] I <= zero_extend[imm]	Sign/Zero' = 0	2
	Reg/Imm' = 0 ALUop[3:0] = 1101 sv = x MAR_write = 0	3
S <= A or I	Dmem_write = 0	
reg[rt] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 0 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

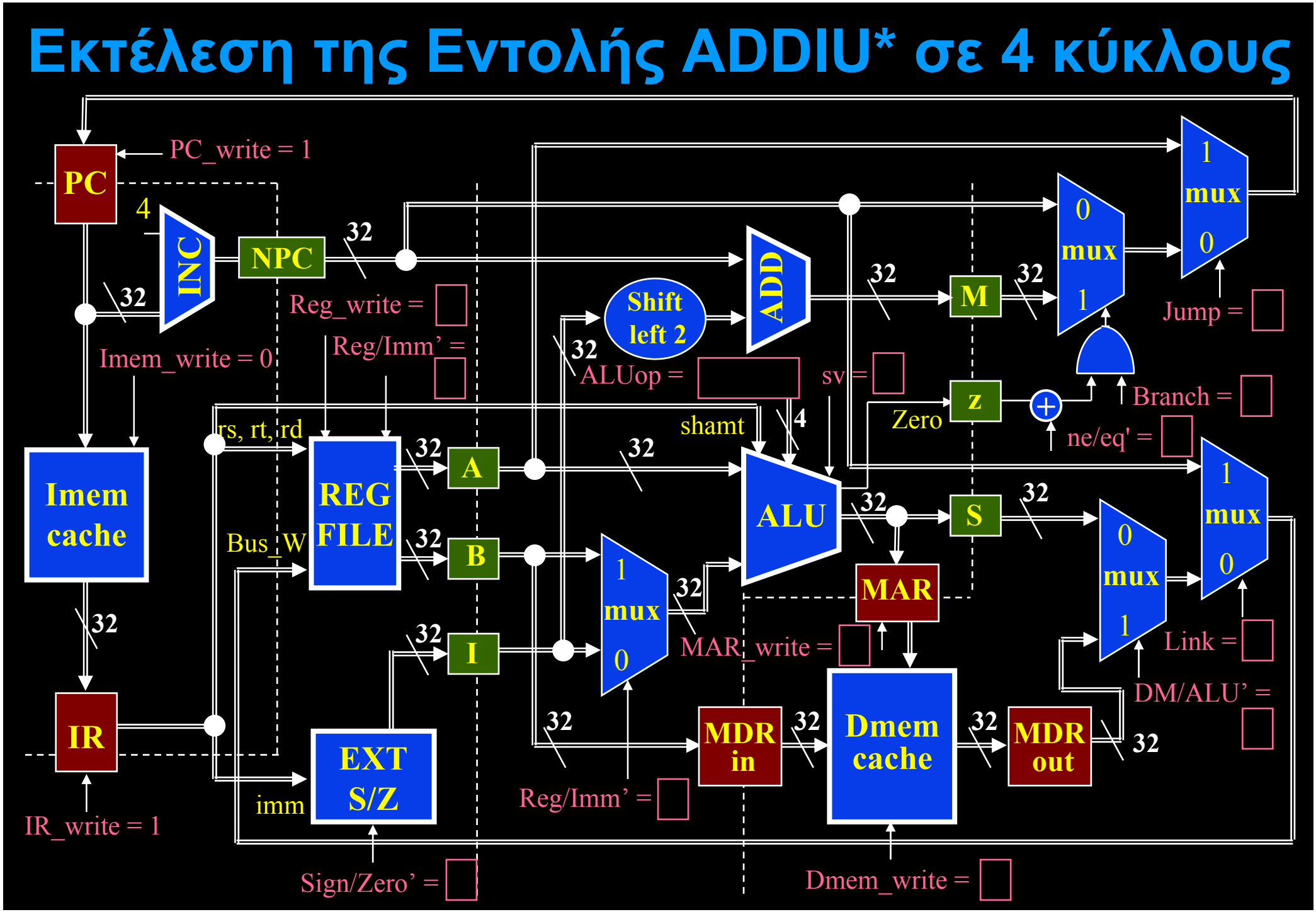
Μικρο-λειτουργίες και Σήματα Ελέγχου της XORI



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] I <= zero_extend[imm]	Sign/Zero' = 0	2
	Reg/Imm' = 0 ALUop[3:0] = 1110 sv = x MAR_write = 0	3
S <= A xor I	Dmem_write = 0	
reg[rt] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 0 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

Εκτέλεση της Εντολής ADDIU* σε 4 κύκλους

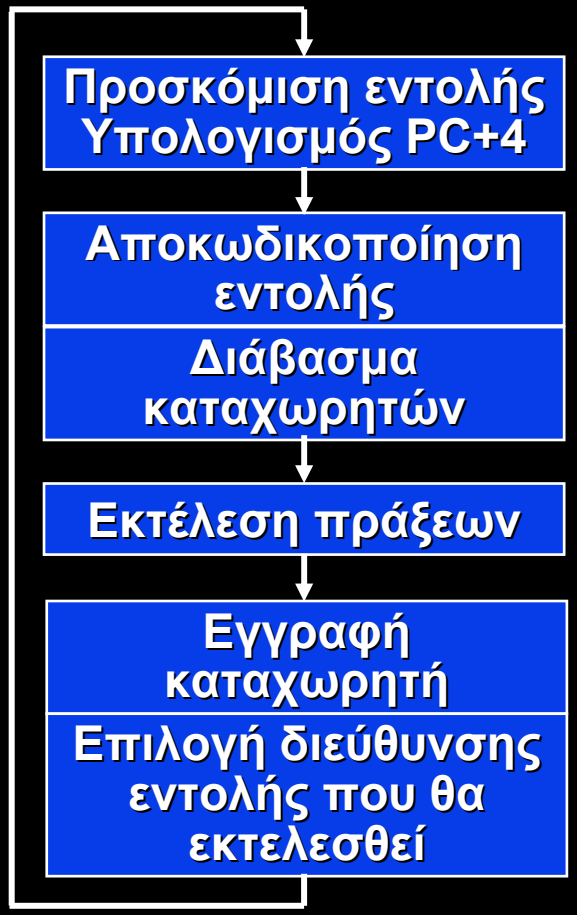


Εκτέλεση της Εντολής ADDU* σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Διάβασμα του καταχωρητή **rt** από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Εκτέλεση της πράξης (+/-/and/or/xor/nor) στην ALU
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή **rd** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

* Παρόμοια οι εντολές SUBU, AND, OR, XOR, NOR

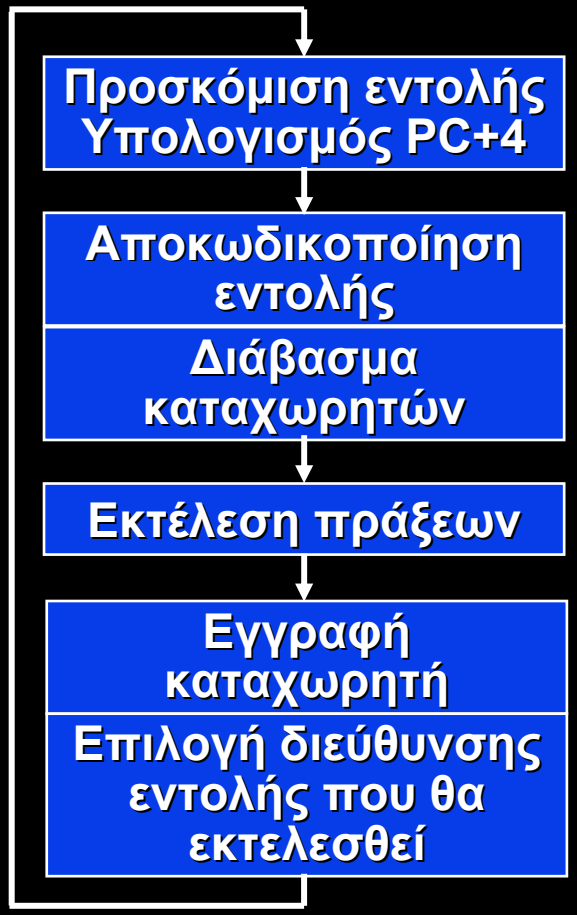
Μικρο-λειτουργίες και Σήματα Ελέγχου της ADDU



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 1001 sv = x MAR_write = 0	3
S <= A + B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

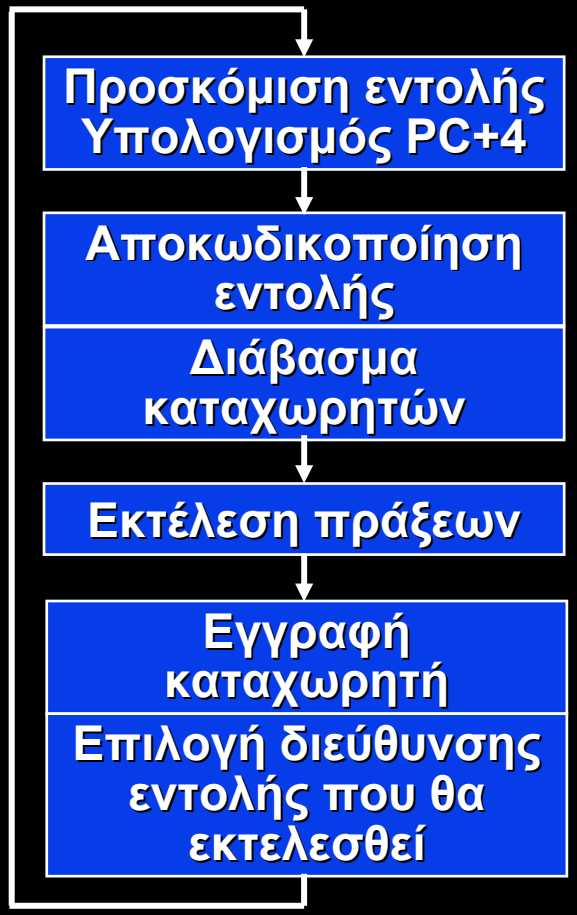
Μικρο-λειτουργίες και Σήματα Ελέγχου της SUBU



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 1011 sv = x MAR_write = 0	3
S <= A - B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

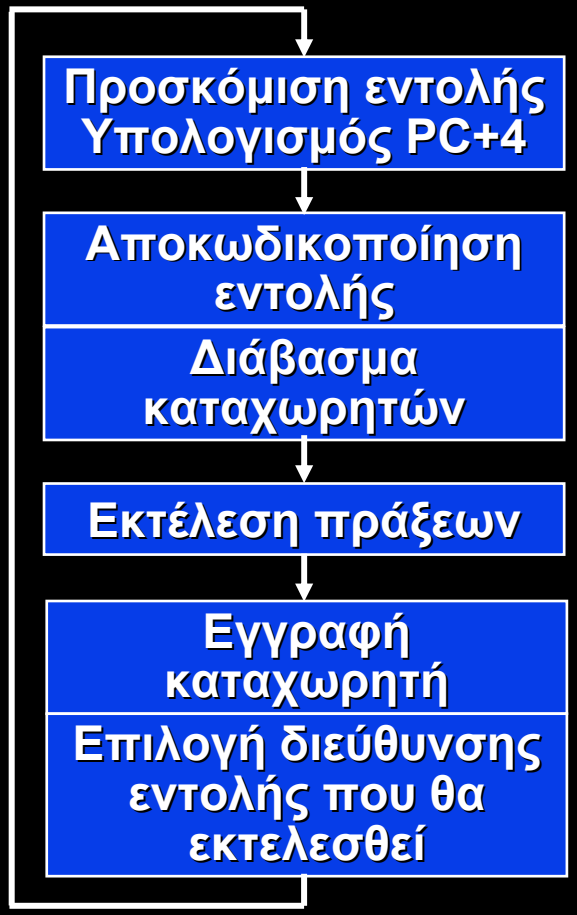
Μικρο-λειτουργίες και Σήματα Ελέγχου της AND



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 1100 sv = x MAR_write = 0	3
S <= A and B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

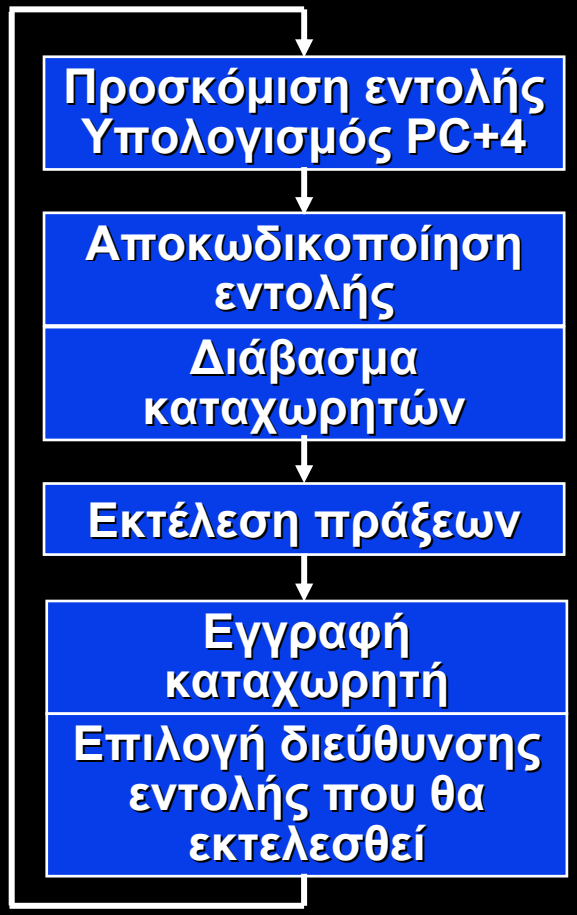
Μικρο-λειτουργίες και Σήματα Ελέγχου της OR



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 1101 sv = x MAR_write = 0	3
S <= A or B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

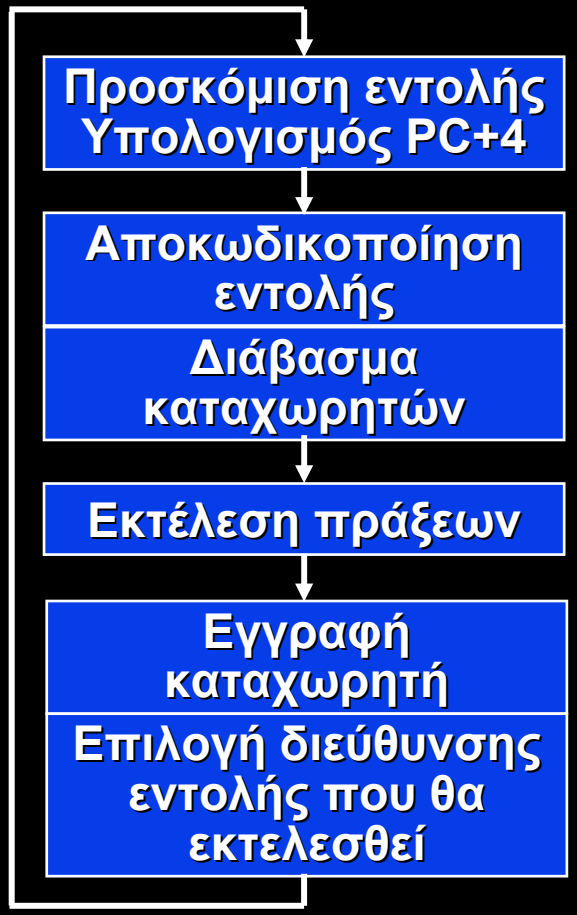
Μικρο-λειτουργίες και Σήματα Ελέγχου της XOR



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 1110 sv = x MAR_write = 0	3
S <= A xor B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

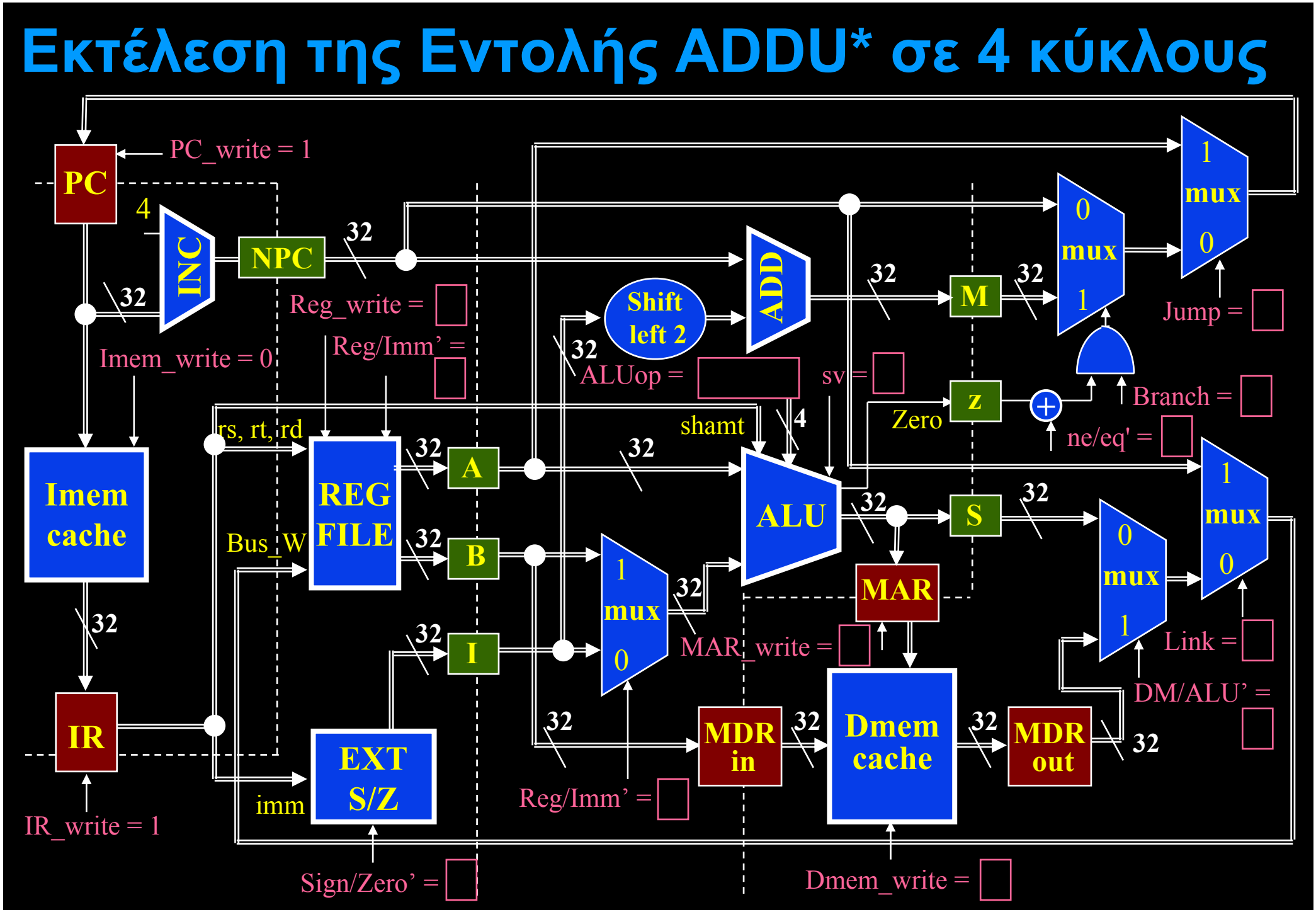
Μικρο-λειτουργίες και Σήματα Ελέγχου της NOR



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero’ = x	2
	Reg/Imm’ = 1 ALUop[3:0] = 1111 sv = x MAR_write = 0	3
S <= A nor B		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU’ = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

Εκτέλεση της Εντολής ADDU* σε 4 κύκλους

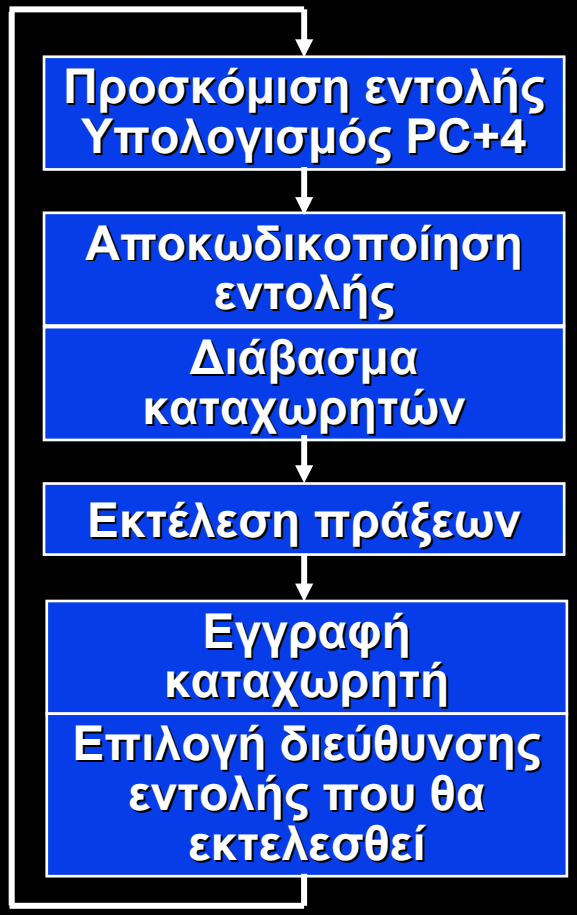


Εκτέλεση της Εντολής SLL* σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή rt από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Εκτέλεση της ολίσθησης (SLL, SRL, SRA) στον rt κατά shamt στην ALU
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή rd του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

* Παρόμοια οι εντολές SRL και SRA

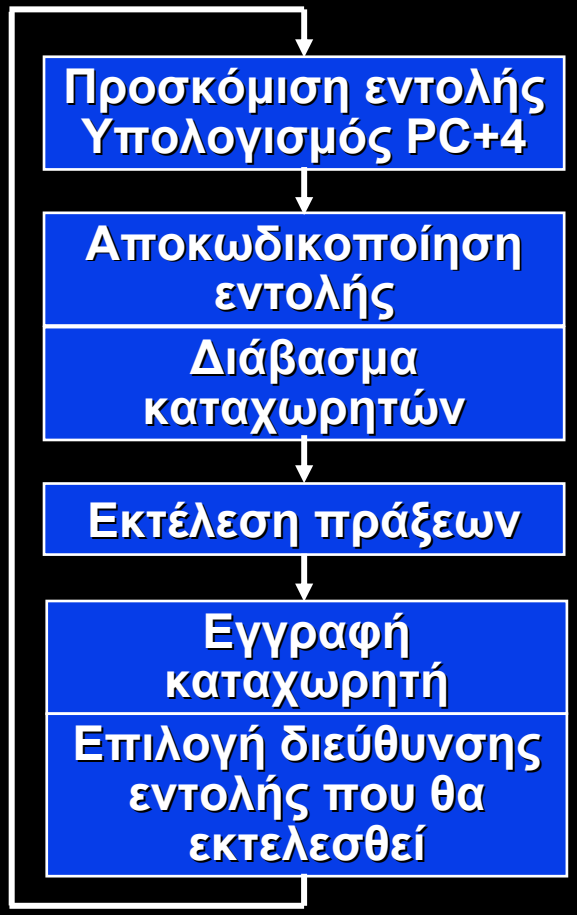
Μικρο-λειτουργίες και Σήματα Ελέγχου της SLL



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 0000 sv = 0 MAR_write = 0	3
S <= B SLL by shamt		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

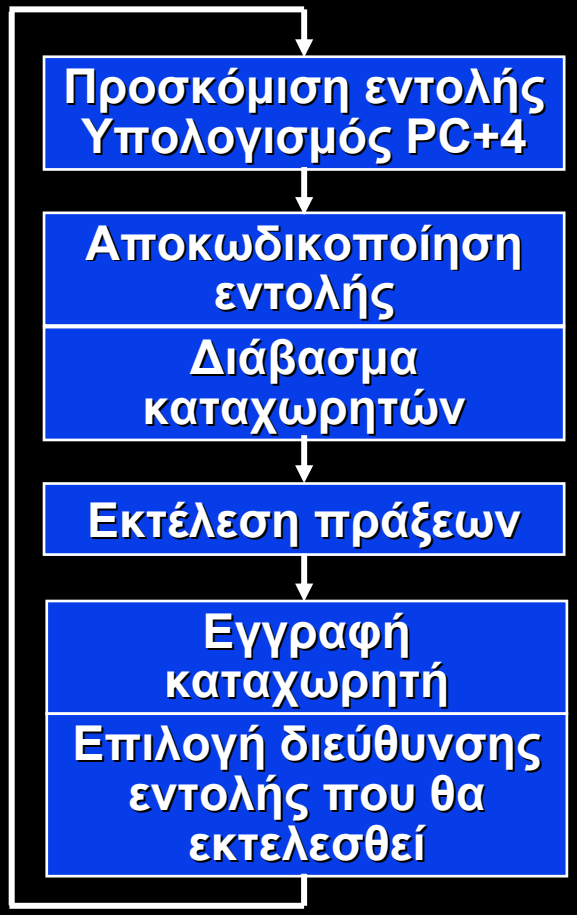
Μικρο-λειτουργίες και Σήματα Ελέγχου της SRL



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
B <= reg[rt]	Sign/Zero’ = x	2
	Reg/Imm’ = 1 ALUop[3:0] = 0010 sv = 0 MAR_write = 0	3
S <= B SRL by shamt		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU’ = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

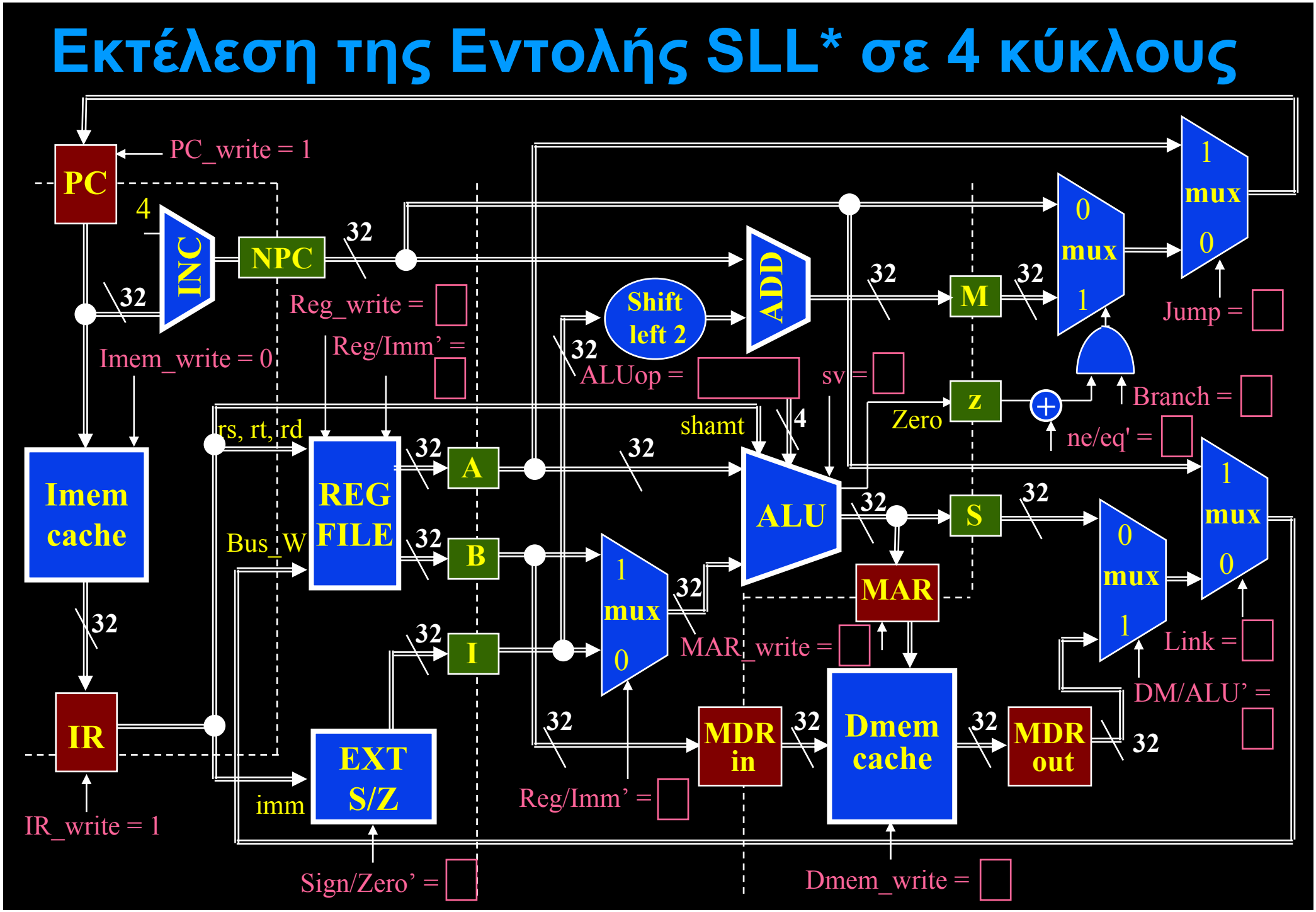
Μικρο-λειτουργίες και Σήματα Ελέγχου της SRA



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
B <= reg[rt]	Sign/Zero’ = x	2
	Reg/Imm’ = 1 ALUop[3:0] = 0011 sv = 0 MAR_write = 0	3
S <= B SRA by shamt		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU’ = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

Εκτέλεση της Εντολής SLL* σε 4 κύκλους

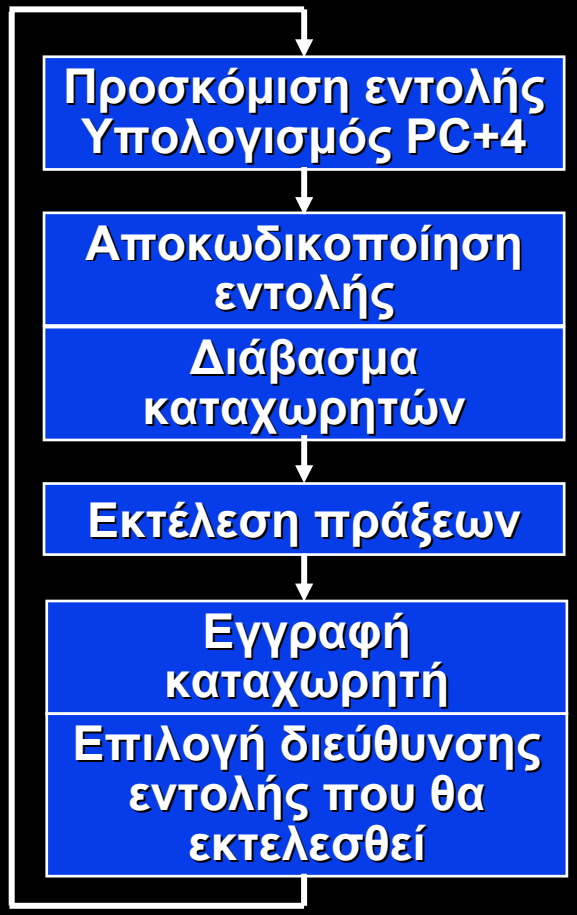


Εκτέλεση της Εντολής SLLV* σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Διάβασμα του καταχωρητή **rt** από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Εκτέλεση ολίσθησης (SLL, SRL, SRA) στον **rt** κατά **rs** στην ALU
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή **rd** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

* Παρόμοια οι εντολές SRLV και SRAV

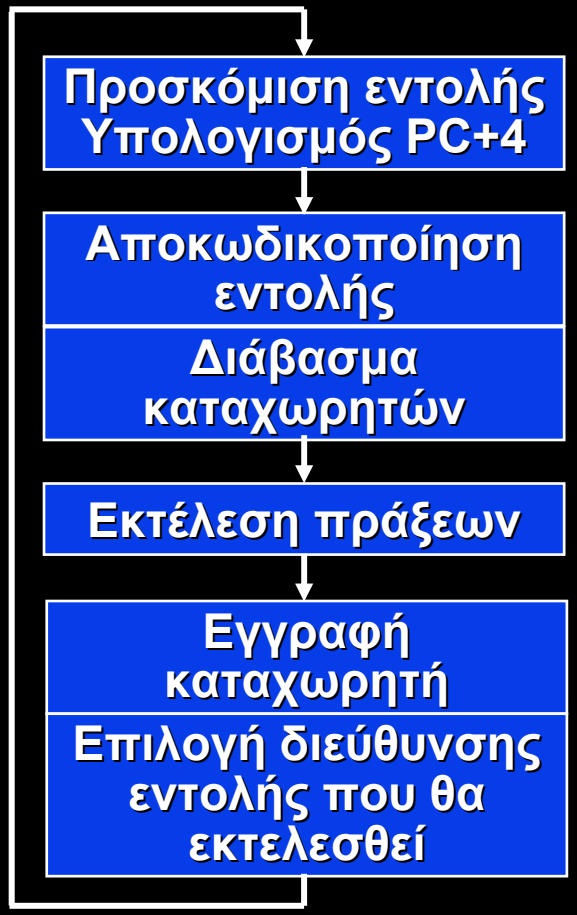
Μικρο-λειτουργίες και Σήματα Ελέγχου της SLLV



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 0000 sv = 1 MAR_write = 0	3
S <= B SLL by A		
reg[rd] <= S PC <= NPC	Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

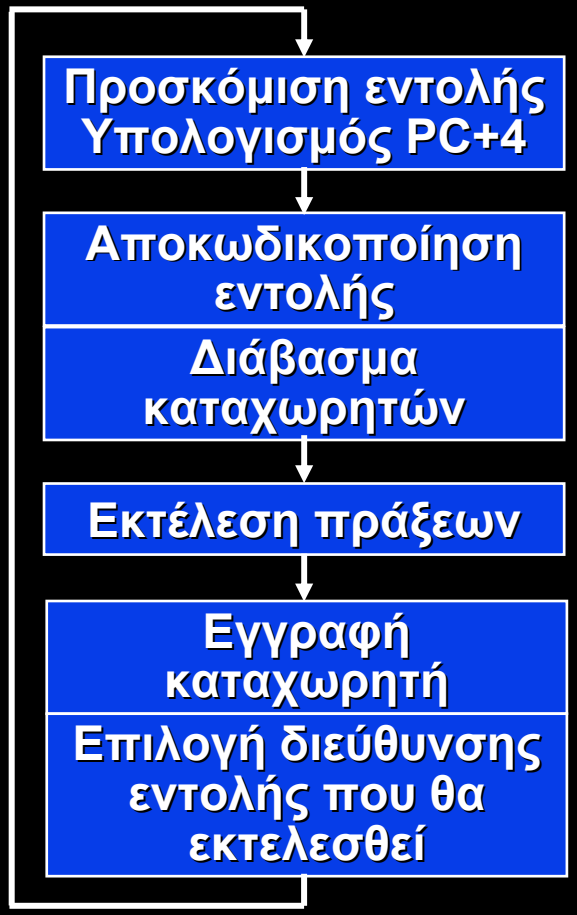
Μικρο-λειτουργίες και Σήματα Ελέγχου της SRLV



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 0010 sv = 1 MAR_write = 0	3
S <= B SRL by A	Dmem_write = 0	
reg[rd] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

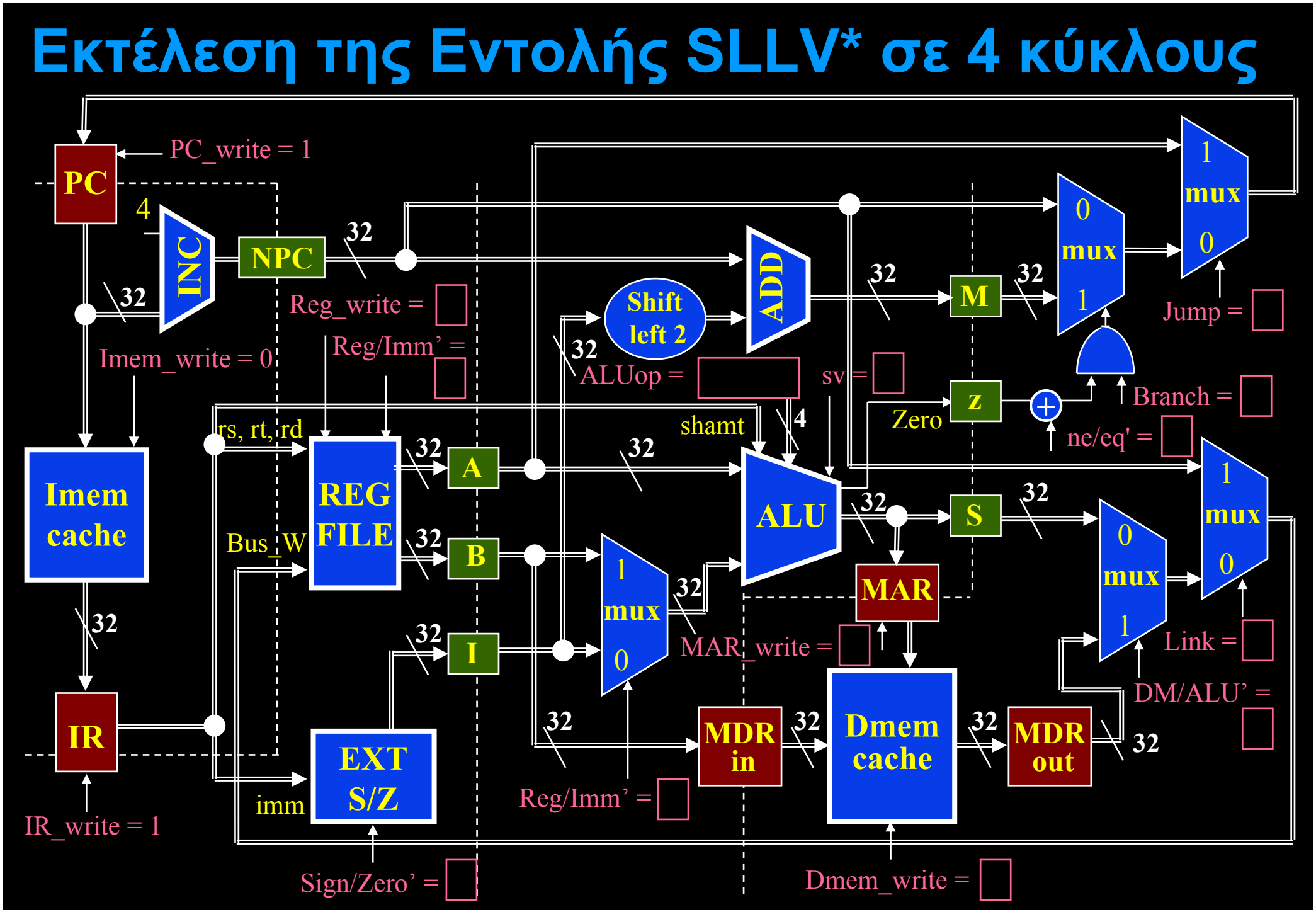
Μικρο-λειτουργίες και Σήματα Ελέγχου της SRAV



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs] B <= reg[rt]	Sign/Zero' = x	2
	Reg/Imm' = 1 ALUop[3:0] = 0011 sv = 1 MAR_write = 0	3
S <= B SRA by A	Dmem_write = 0	
reg[rd] <= S PC <= NPC	Reg_write = 1 Reg/Imm' = 1 DM/ALU' = 0 Link = 0 Jump = 0 Branch = 0 ne/eq' = x PC_write = 1	4

Imem_en = Dmem_en = 1 πάντα

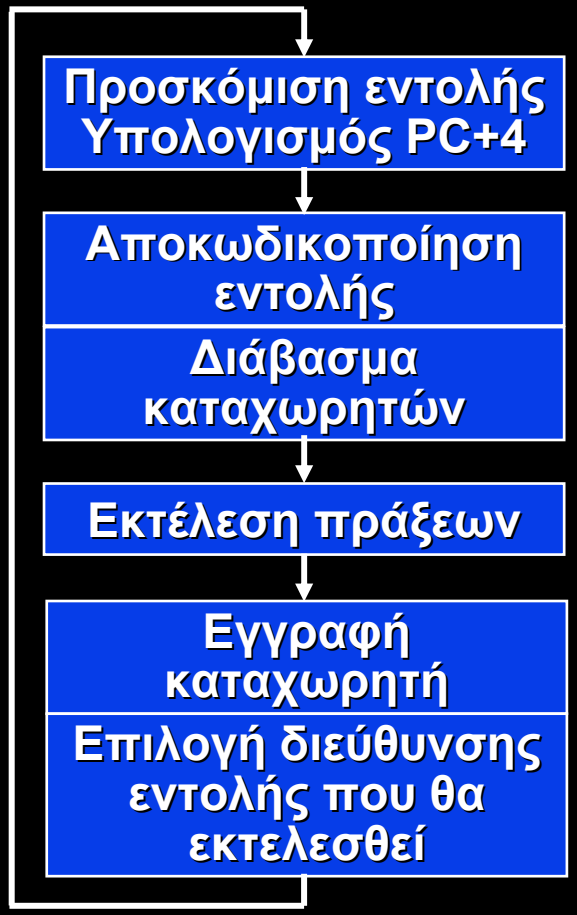
Εκτέλεση της Εντολής SLLV* σε 4 κύκλους



Εκτέλεση της Εντολής SLTI σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Επέκταση πρόσημου του πεδίου **immediate**
- **3ος κύκλος**
  - Εκτέλεση της σύγκρισης (**rs < sign_extend[imm]**) σαν αφαίρεση στην ALU και εμφάνιση του πρόσημου του αποτελέσματος
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή **rt** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

Μικρο-λειτουργίες και Σήματα Ελέγχου της SLTI




```
IR <= Imem[PC]
NPC <= PC + 4
```

```
A <= reg[rs]
I <=sign_extend[imm]
```

```
If A < I then S <= 1
         else S <= 0
```

```
reg[rt] <= S
PC <= NPC
```

```
Imem_write = 0
IR_write = 1
```

1

```
Sign/Zero’ = 1
```

2

```
Reg/Imm’ = 0
ALUop[3:0] = 0110
sv = x
MAR_write = 0
```

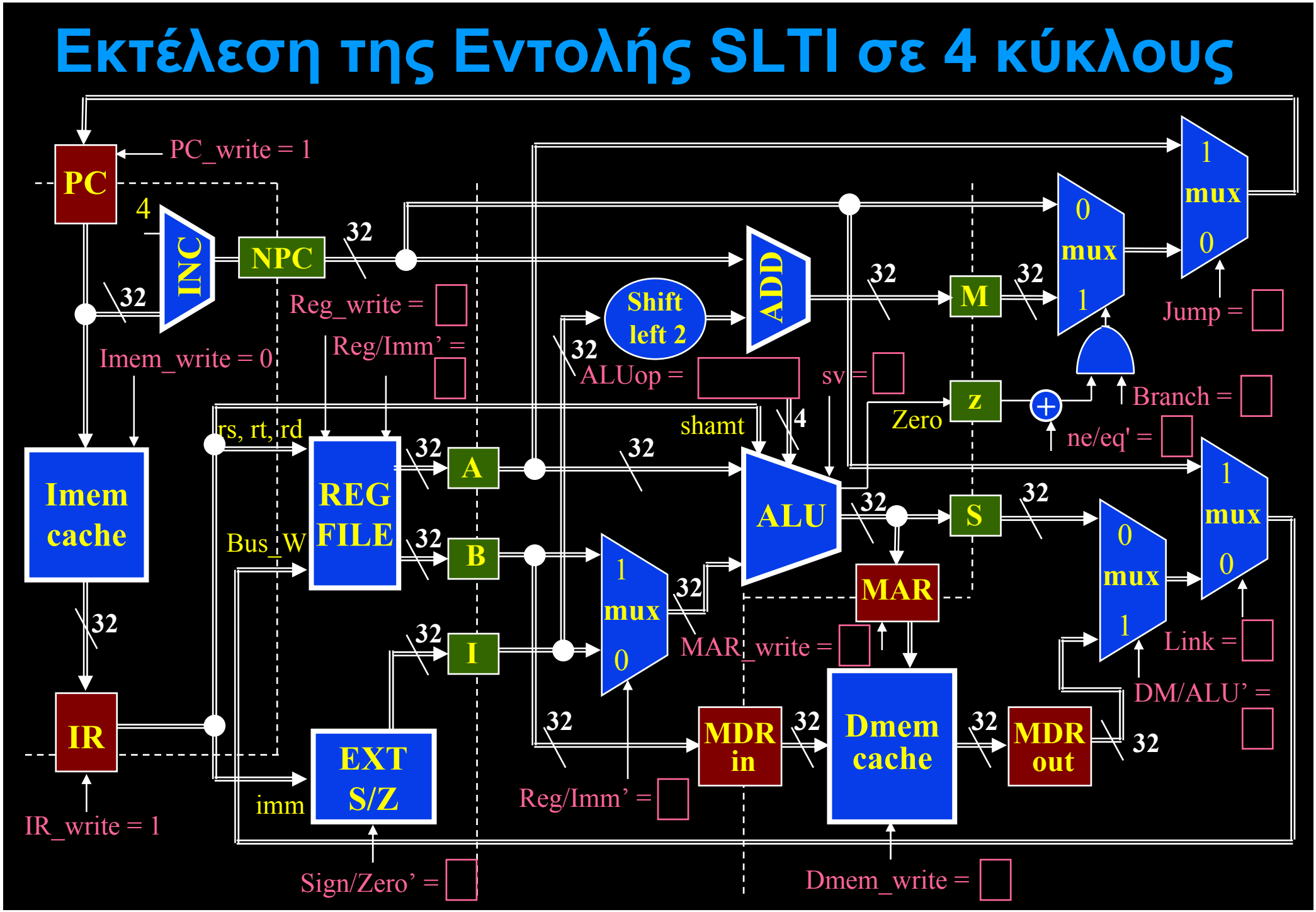
3

```
Dmem_write = 0
Reg_write = 1
Reg/Imm' = 0
DM/ALU’ = 0
Link = 0
Jump = 0
Branch = 0
ne/eq' = x
PC_write = 1
```

4

```
Imem_en = Dmem_en = 1  πάντα
```

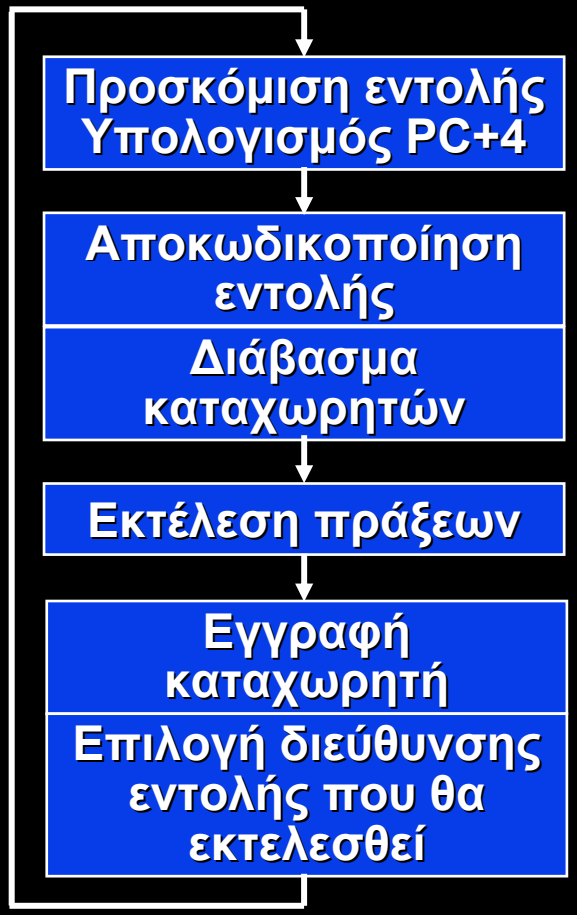
Εκτέλεση της Εντολής SLTI σε 4 κύκλους



Εκτέλεση της Εντολής SLT σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
  - Διάβασμα του καταχωρητή **rt** από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Εκτέλεση της σύγκρισης (**rs < rt**) σαν αφαίρεση στην ALU και εμφάνιση του πρόσημου του αποτελέσματος
- **4ος κύκλος**
  - Εγγραφή στον καταχωρητή **rd** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (PC + 4)

Μικρο-λειτουργίες και Σήματα Ελέγχου της SLT




```
IR <= Imem[PC]
NPC <= PC + 4
```

```
A <= reg[rs]
B <= reg[rt]
```

```
If A < B then S <= 1
         else S <= 0
```

```
reg[rd] <= S
PC <= NPC
```

```
Imem_write = 0
IR_write = 1
```

1

```
Sign/Zero' = x
```

2

```
Reg/Imm' = 1
ALUop[3:0] = 0110
sv = x
MAR_write = 0
```

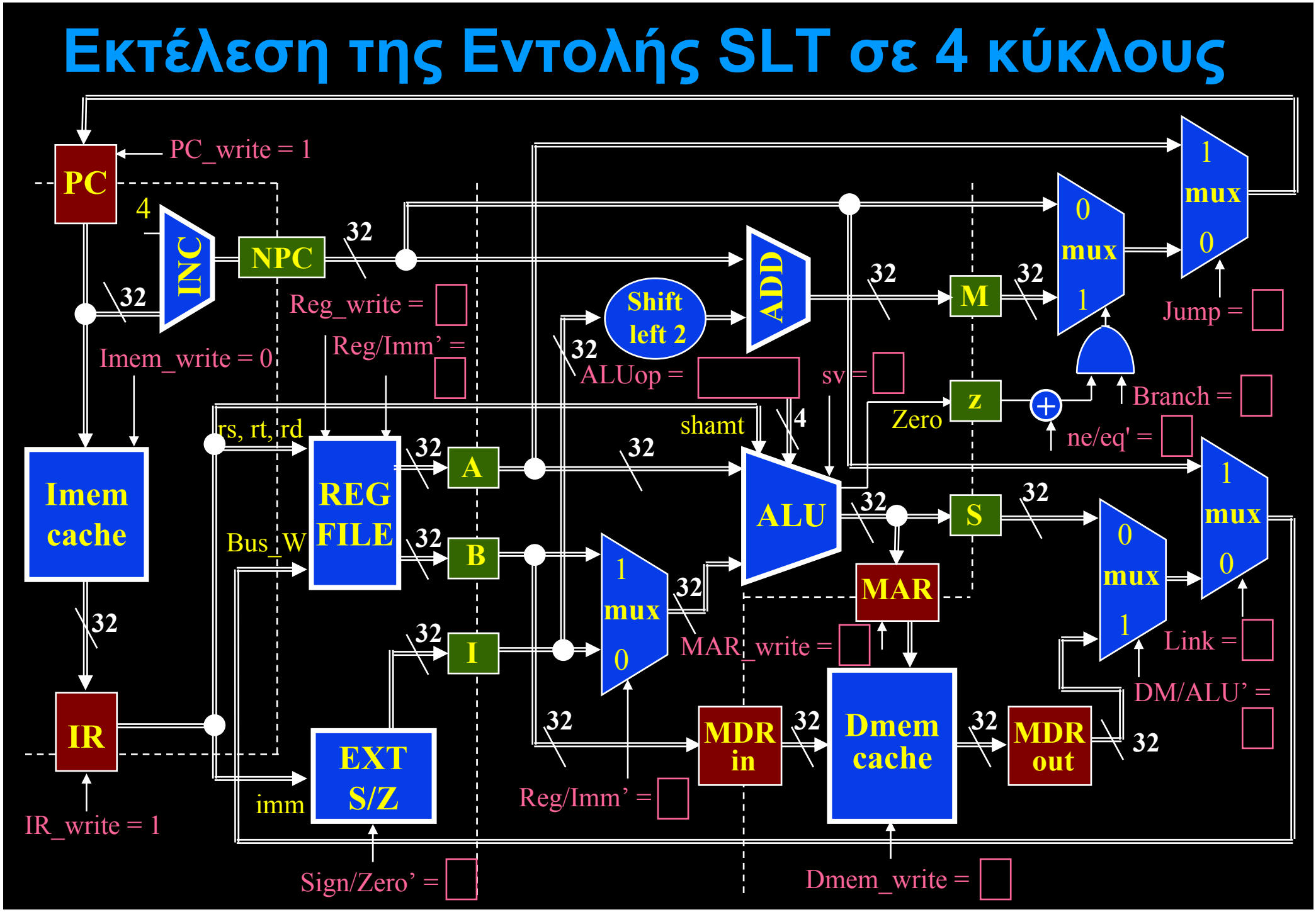
3

```
Dmem_write = 0
Reg_write = 1
Reg/Imm' = 1
DM/ALU' = 0
Link = 0
Jump = 0
Branch = 0
ne/eq' = x
PC_write = 1
```

4

```
Imem_en = Dmem_en = 1  πάντα
```

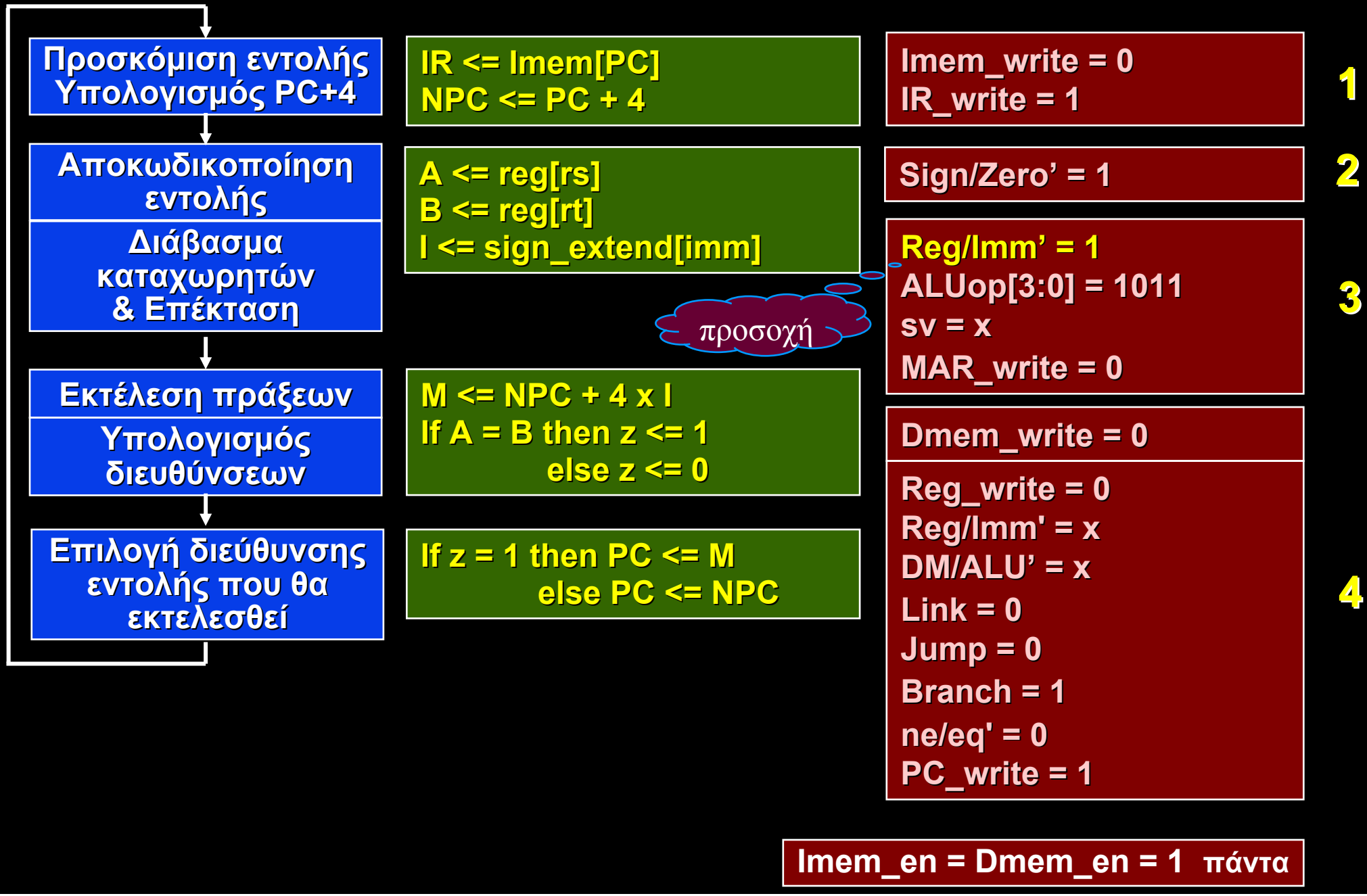
Εκτέλεση της Εντολής SLT σε 4 κύκλους



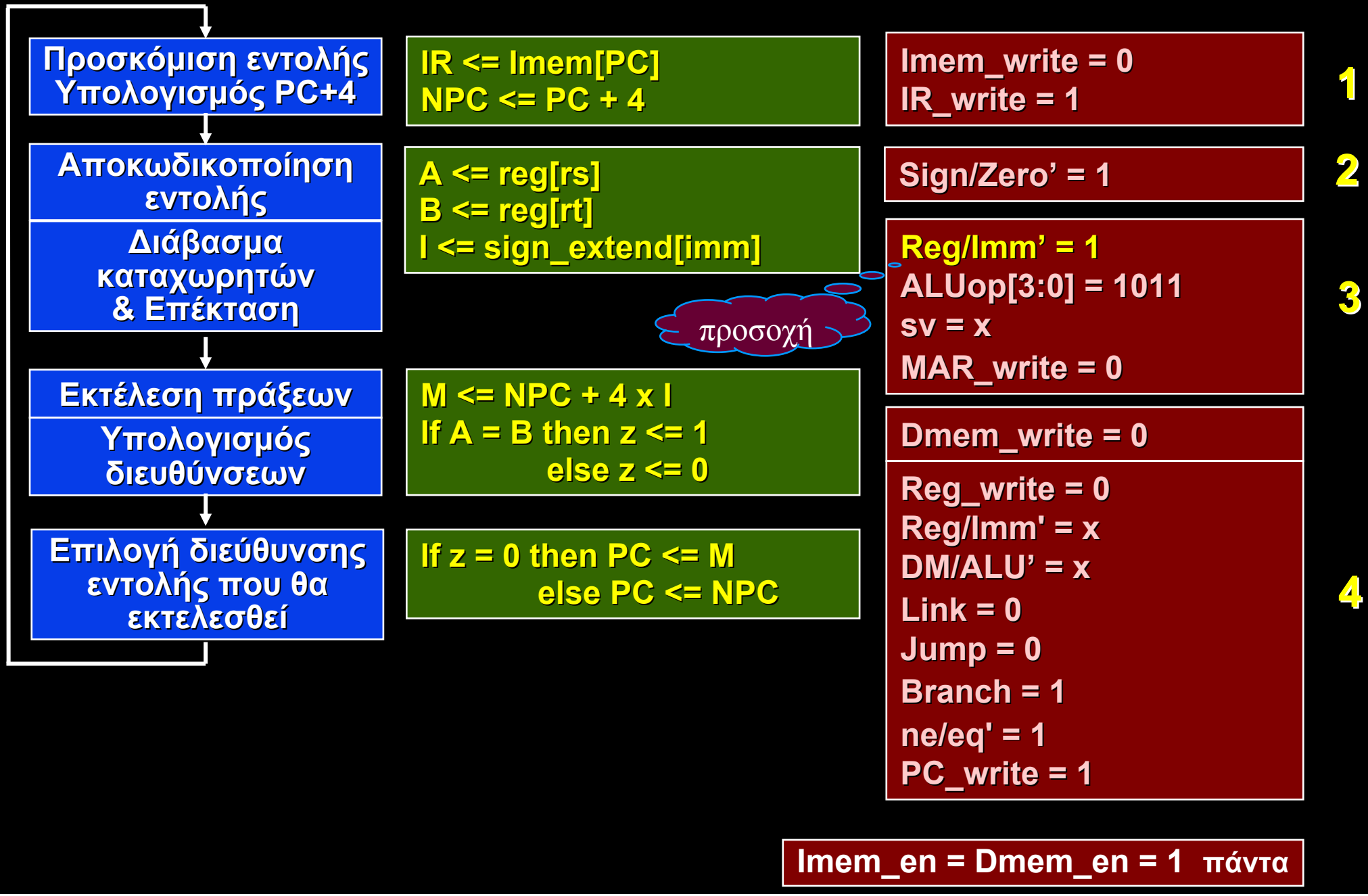
Εκτέλεση της Εντολής BEQ/BNE σε 4 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή rs από το Αρχείο Καταχωρητών
  - Διάβασμα του καταχωρητή rt από το Αρχείο Καταχωρητών
  - Επέκταση πρόσημου του πεδίου immediate που περιέχει σαν μετατόπιση τη διαφορά σε εντολές μ
- **3ος κύκλος**
  - Μετατροπή της διαφοράς σε εντολές μ σε διαφορά σε bytes 4μ με αριστερή ολίσθηση κατά 2 ψηφία (πολλαπλασιασμός επί 4)
  - Υπολογισμός της διεύθυνσης PC + 4 + 4μ
  - Εκτέλεση της σύγκρισης (rs =/≠ rt) σαν αφαίρεση στην ALU και ενημέρωση του D-FF z, εάν το αποτέλεσμα είναι μηδέν
- **4ος κύκλος**
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί
    (BEQ : If z = 1 then PC = PC + 4 + 4μ else PC = PC+4)
    (BNE : If z = 0 then PC = PC + 4 + 4μ else PC = PC+4)

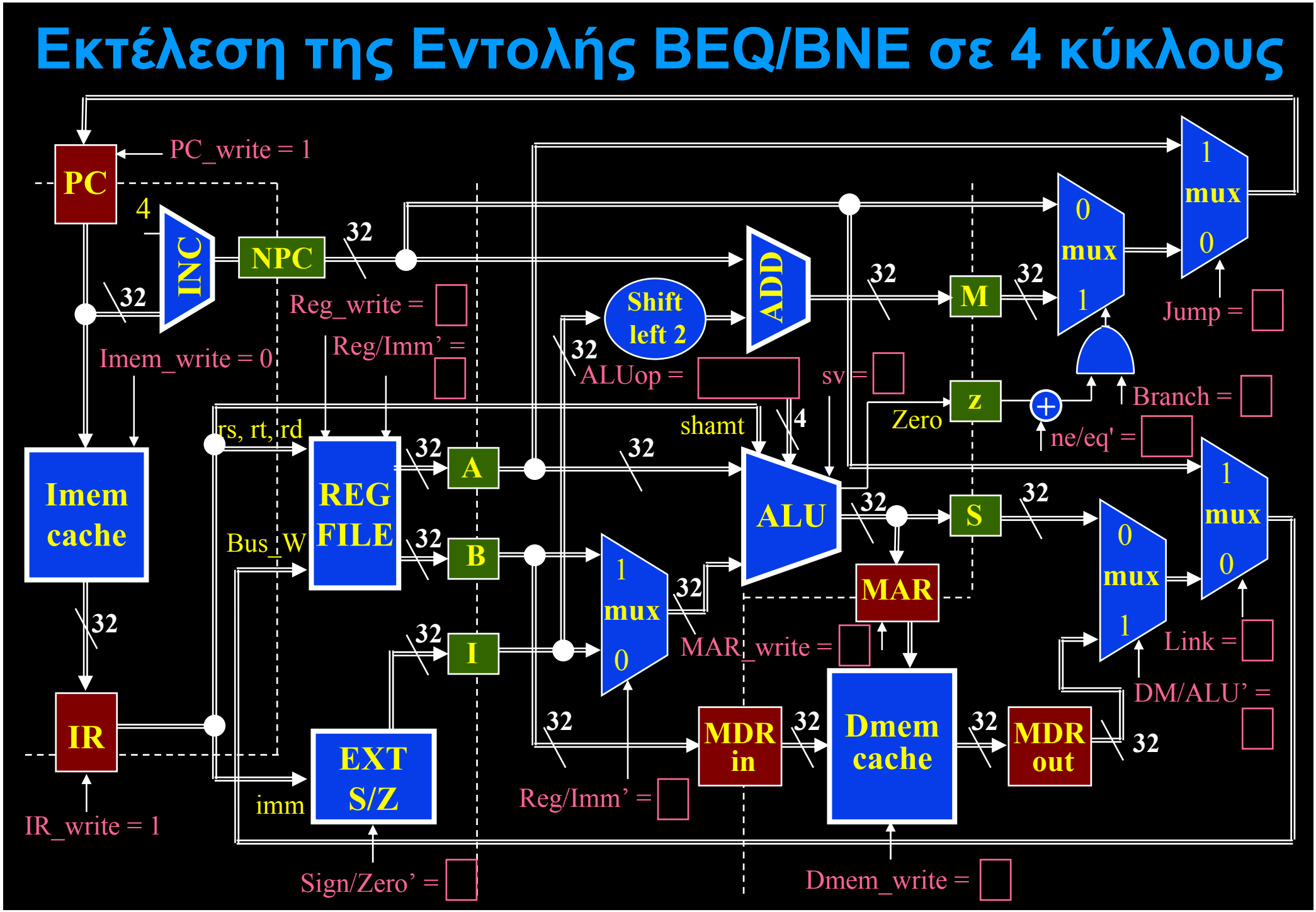
Μικρο-λειτουργίες και Σήματα Ελέγχου της BEQ



Μικρο-λειτουργίες και Σήματα Ελέγχου της BNE



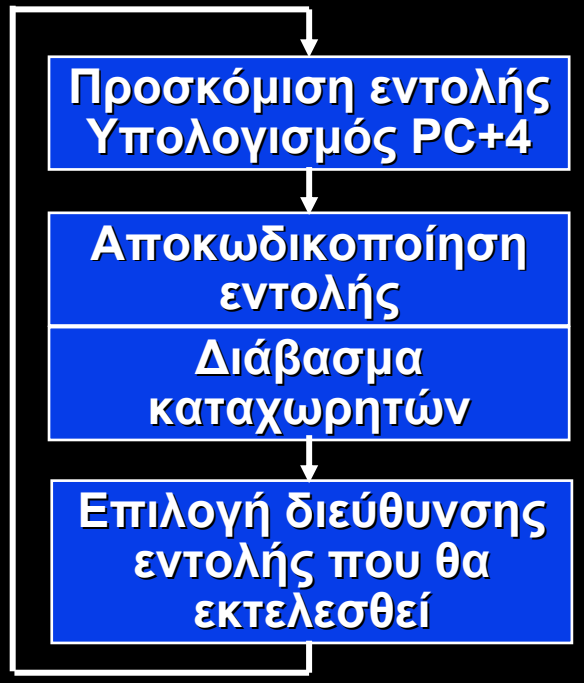
Εκτέλεση της Εντολής BEQ/BNE σε 4 κύκλους



Εκτέλεση της Εντολής JR σε 3 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή rs από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (rs)

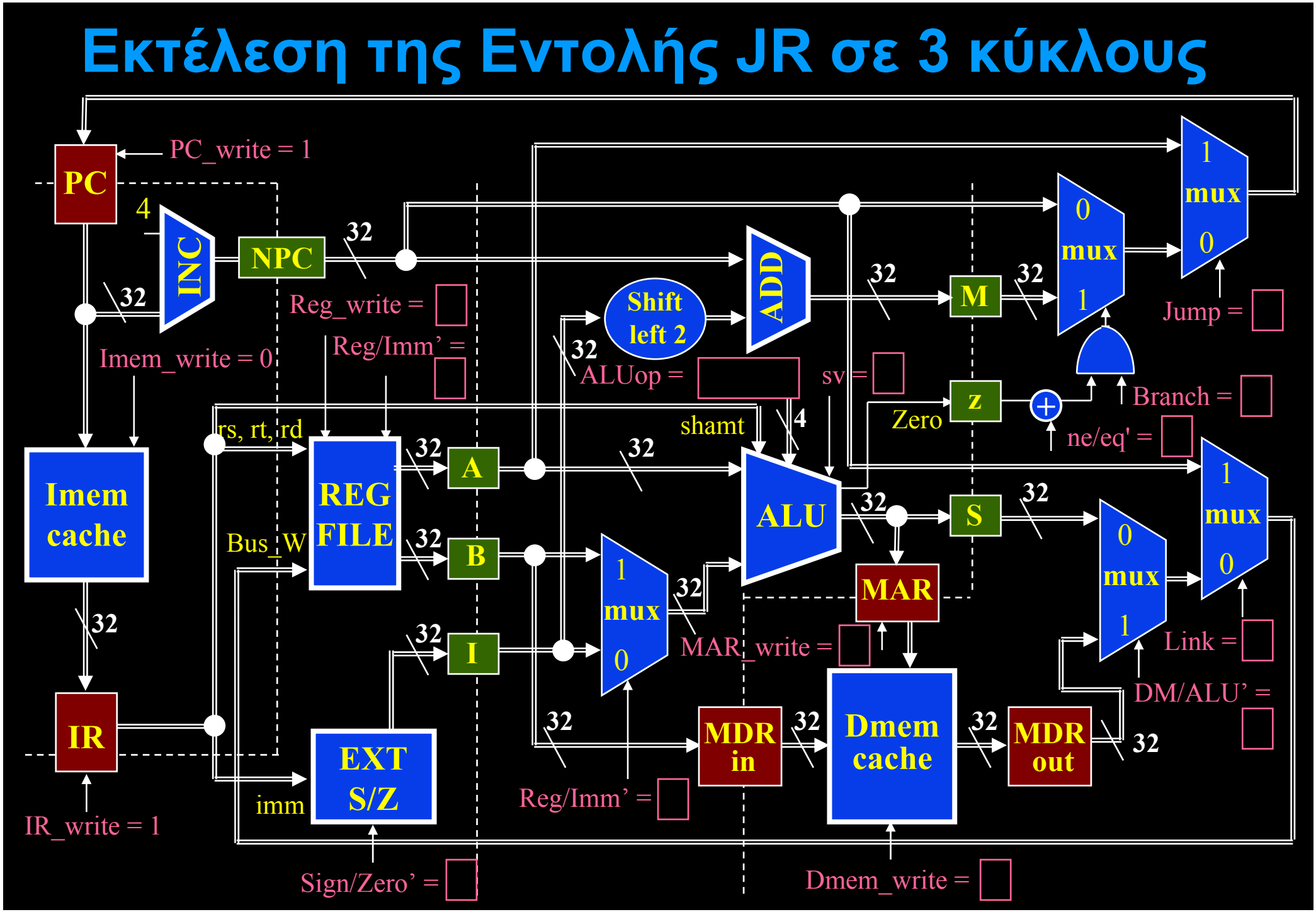
Μικρο-λειτουργίες και Σήματα Ελέγχου της JR



IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1	1
A <= reg[rs]	Sign/Zero’ = x	2
PC <= A	Reg/Imm’ = x ALUop[3:0] = xxxx sv = x MAR_write = 0 Dmem_write = 0 Reg_write = 0 Reg/Imm' = x DM/ALU’ = x Link = x Jump = 1 Branch = x ne/eq' = x PC_write = 1	3

Imem_en = Dmem_en = 1 πάντα

Εκτέλεση της Εντολής JR σε 3 κύκλους

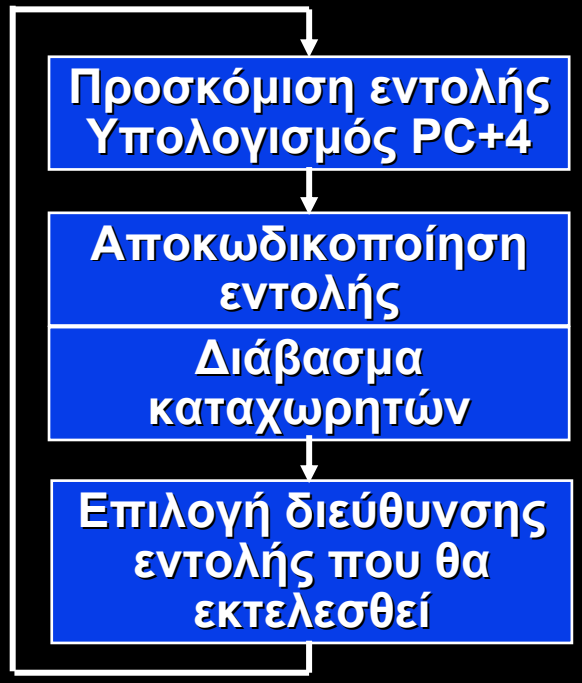


Εκτέλεση της Εντολής JALR σε 3 κύκλους

- **1ος κύκλος**
  - Προσκόμιση της εντολής από τη Μνήμη Εντολών
  - Υπολογισμός της διεύθυνσης της επόμενης εντολής (PC + 4)
- **2ος κύκλος**
  - Αποκωδικοποίηση της εντολής στη Μονάδα Ελέγχου
  - Διάβασμα του καταχωρητή **rs** από το Αρχείο Καταχωρητών
- **3ος κύκλος**
  - Εγγραφή του PC+4 στον καταχωρητή **rd** του Αρχείου Καταχωρητών
  - Επιλογή της διεύθυνσης της εντολής που θα εκτελεσθεί (rs)

Η προκαθορισμένη τιμή (default) του καταχωρητή rd, όταν αυτός δεν δηλώνεται, είναι ο καταχωρητής $ra = $31

Μικρο-λειτουργίες και Σήματα Ελέγχου της JALR



1	IR <= Imem[PC] NPC <= PC + 4	Imem_write = 0 IR_write = 1
2	A <= reg[rs]	Sign/Zero' = x
3	reg[rd] <= NPC PC <= A	Reg/Imm' = x ALUop[3:0] = xxxx sv = x MAR_write = 0 Dmem_write = 0 Reg_write = 1 Reg/Imm' = 1 DM/ALU' = x Link = 1 Jump = 1 Branch = x ne/eq' = x PC_write = 1

Imem_en = Dmem_en = 1 πάντα

Εκτέλεση της Εντολής JALR σε 3 κύκλους