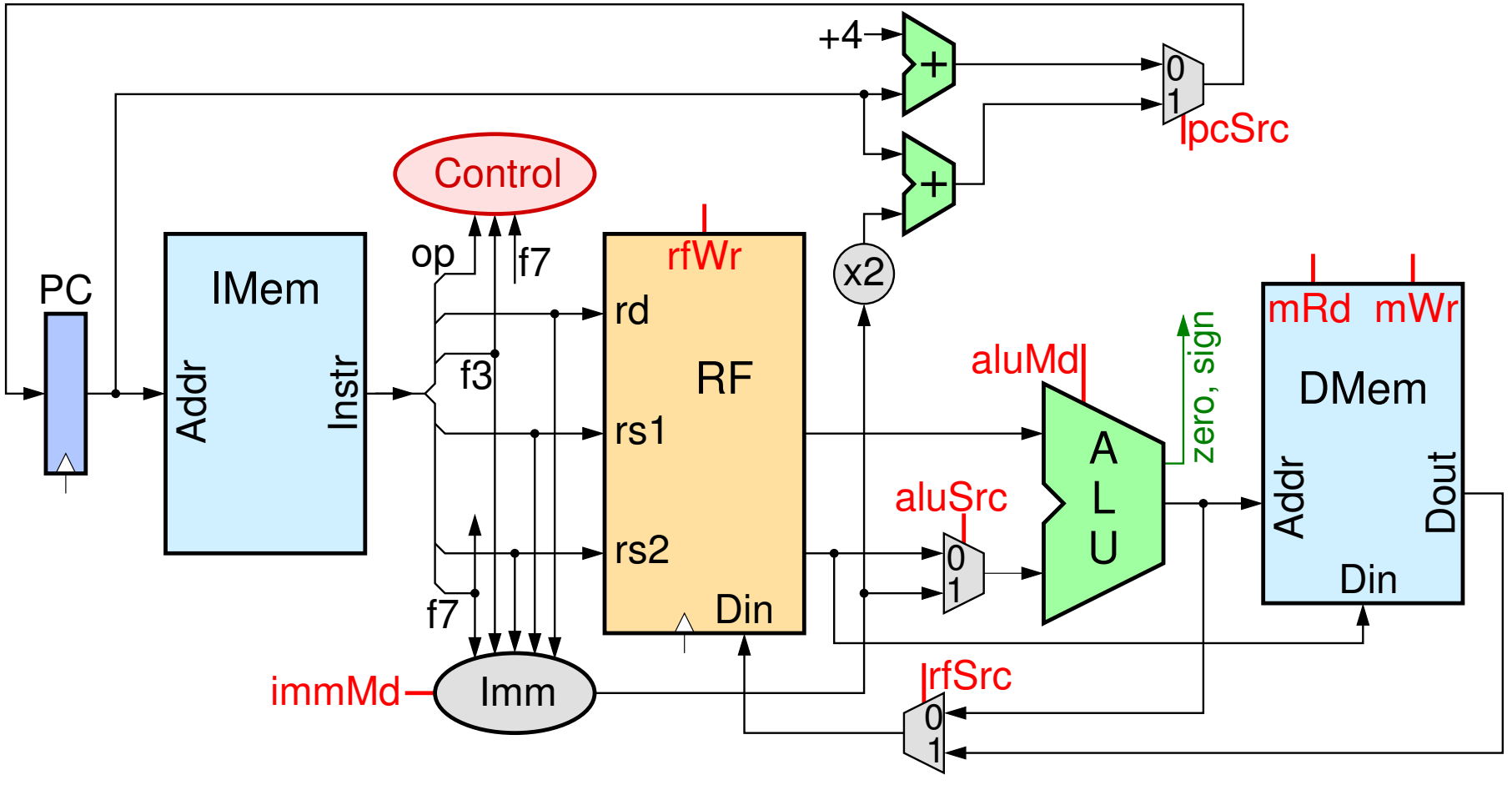
Μία Απλή Υλοποίηση του RISC-V σε έναν (μακρύ) Κύκλο Ρολογιού ανά Εντολή

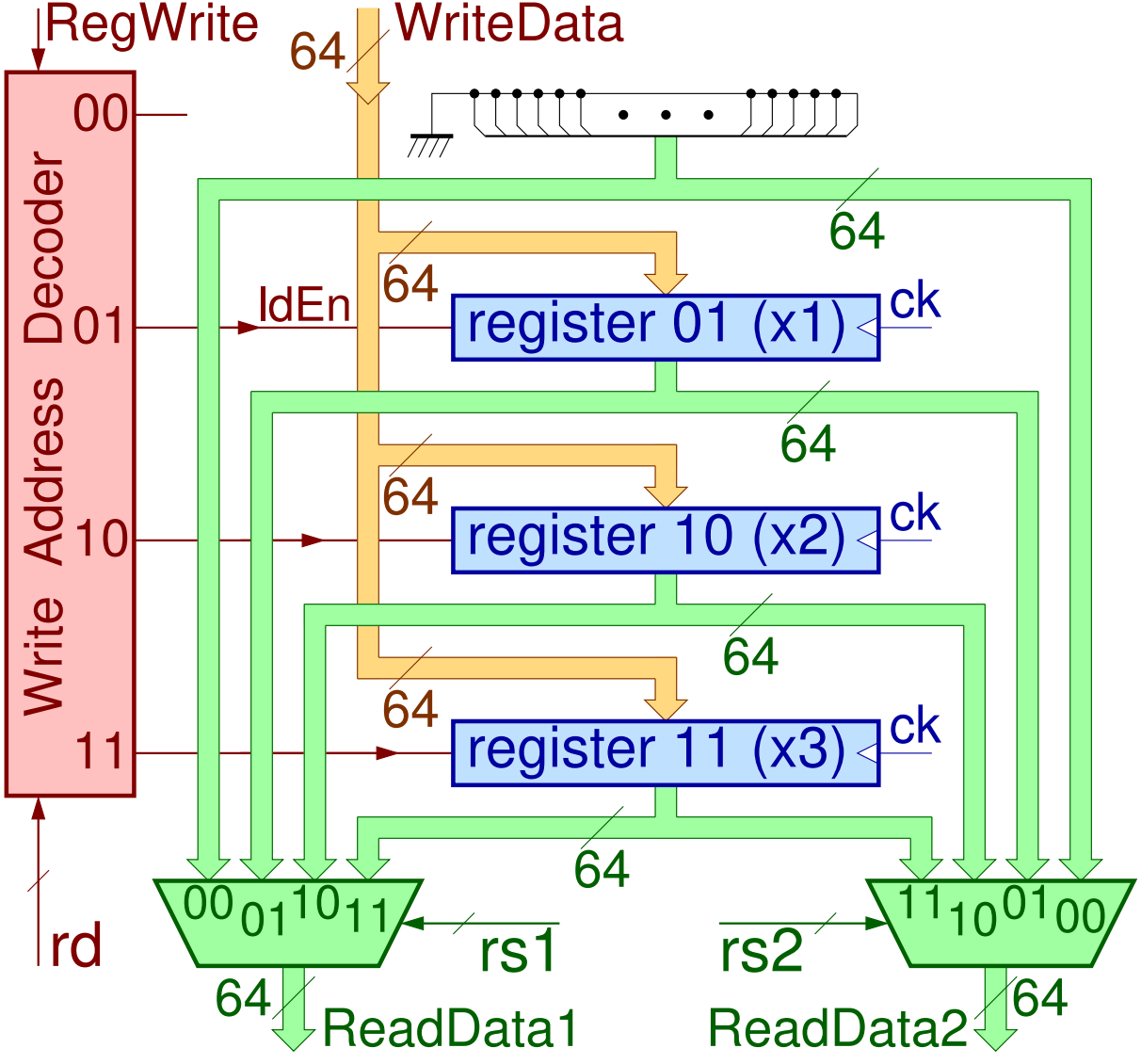
08a (§8.1-8.8) – 19-24 Μαρτίου 2021 – Μανόλης Κατεβαίνης

Διαδρομή Δεδομένων (Datapath) γιά τις βασικές εντ.

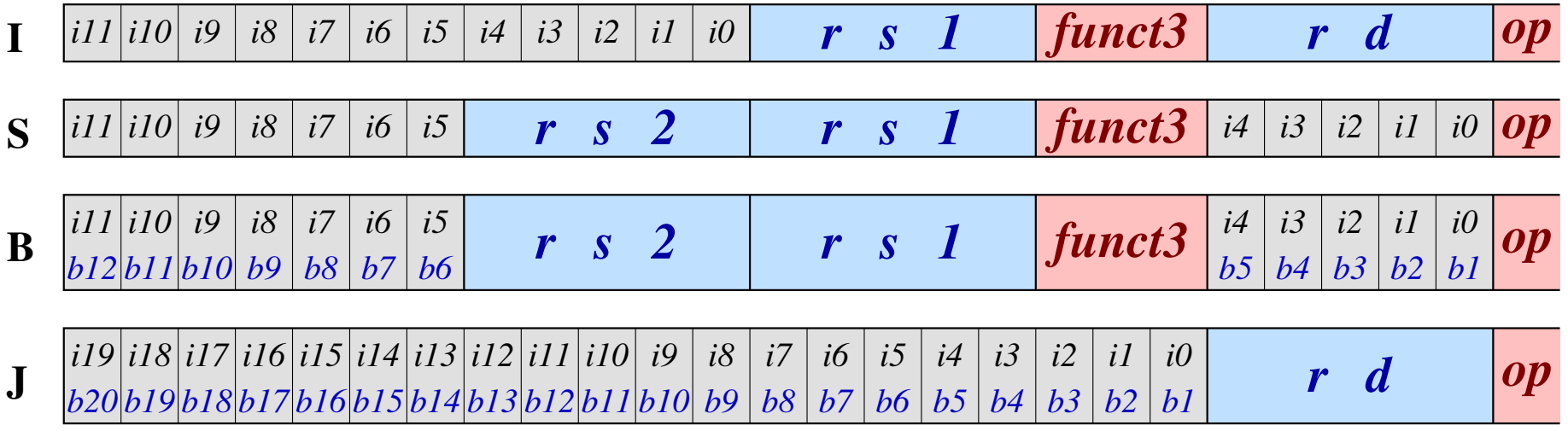


Τρίπορτο Αρχείο Καταχωρητών

- Δίπορτη Ανάγνωση, και
- ταυτόχρονη (μονότ.) Εγγραφή

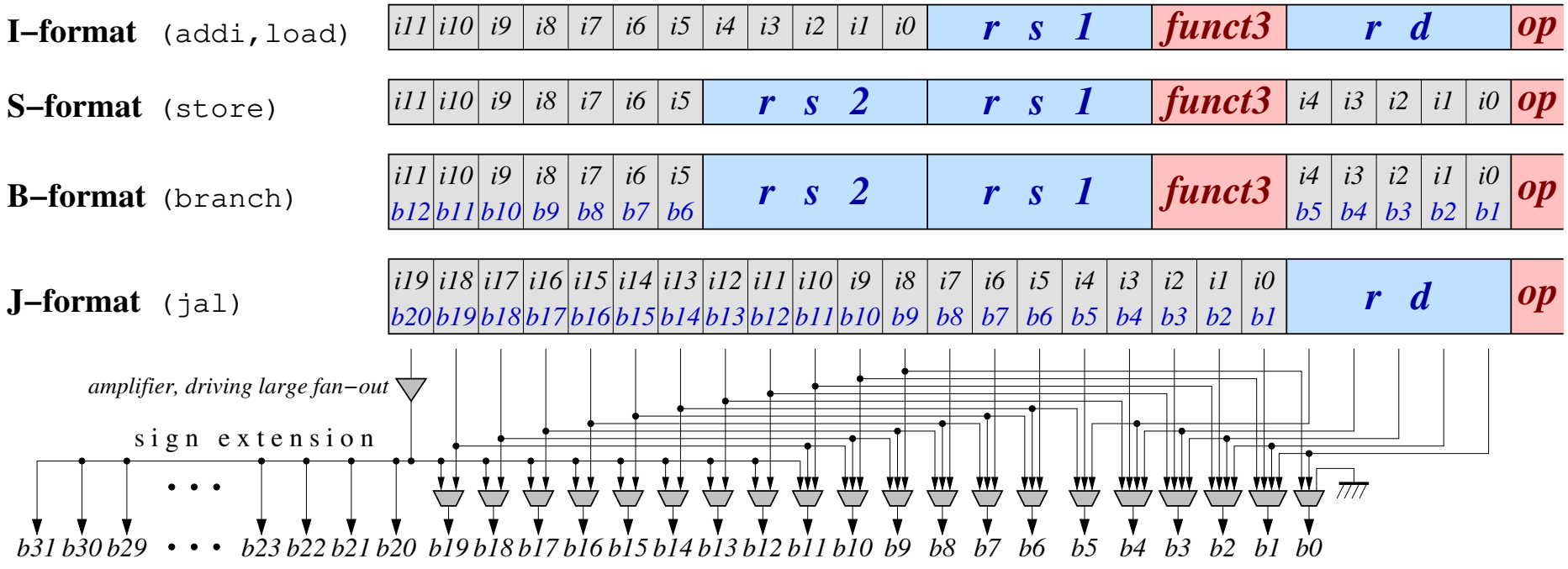


Πού θα περίμενε κανείς τα bits των Immediates



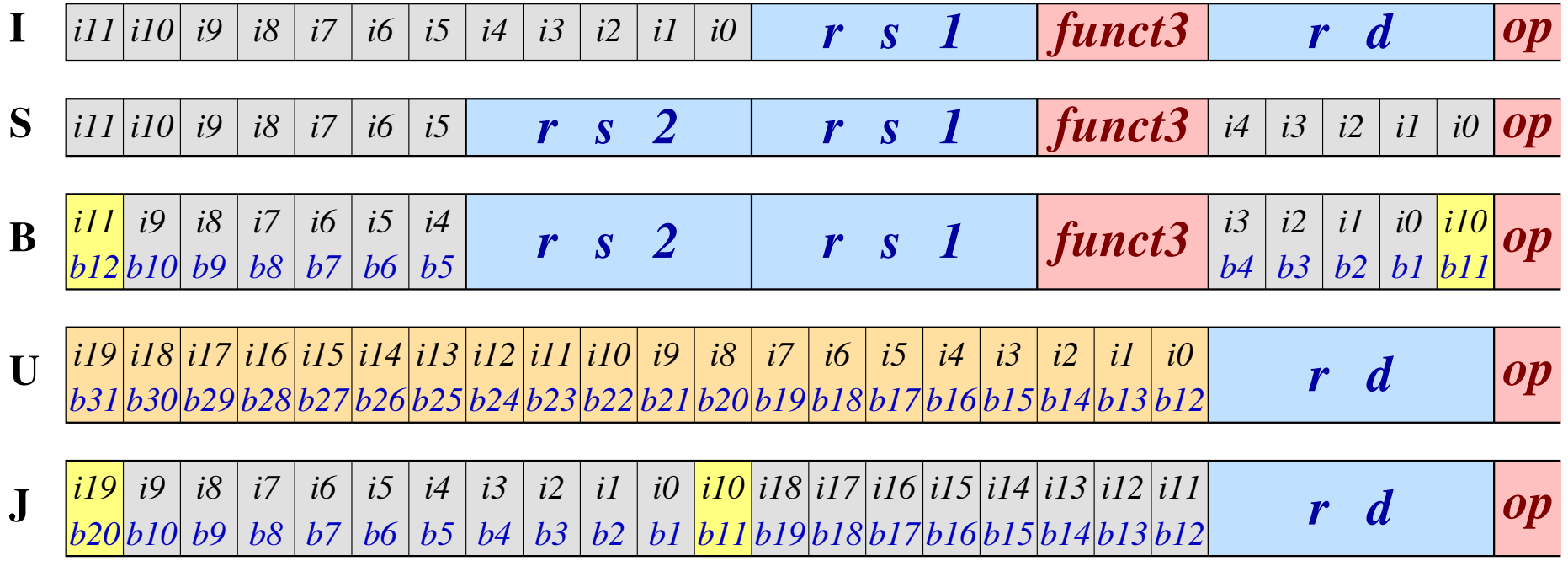
- *i--* η αναμενόμενη αρίθμηση των bits μέσα στην εντολή
- *b--* η θέση που πρέπει να πάνε στον αθροιστή με PC
- Λείπει το U-format (Upper Immediates)

Τι πολυπλέκτες θα απαιτούσαν εάν ήταν έτσι



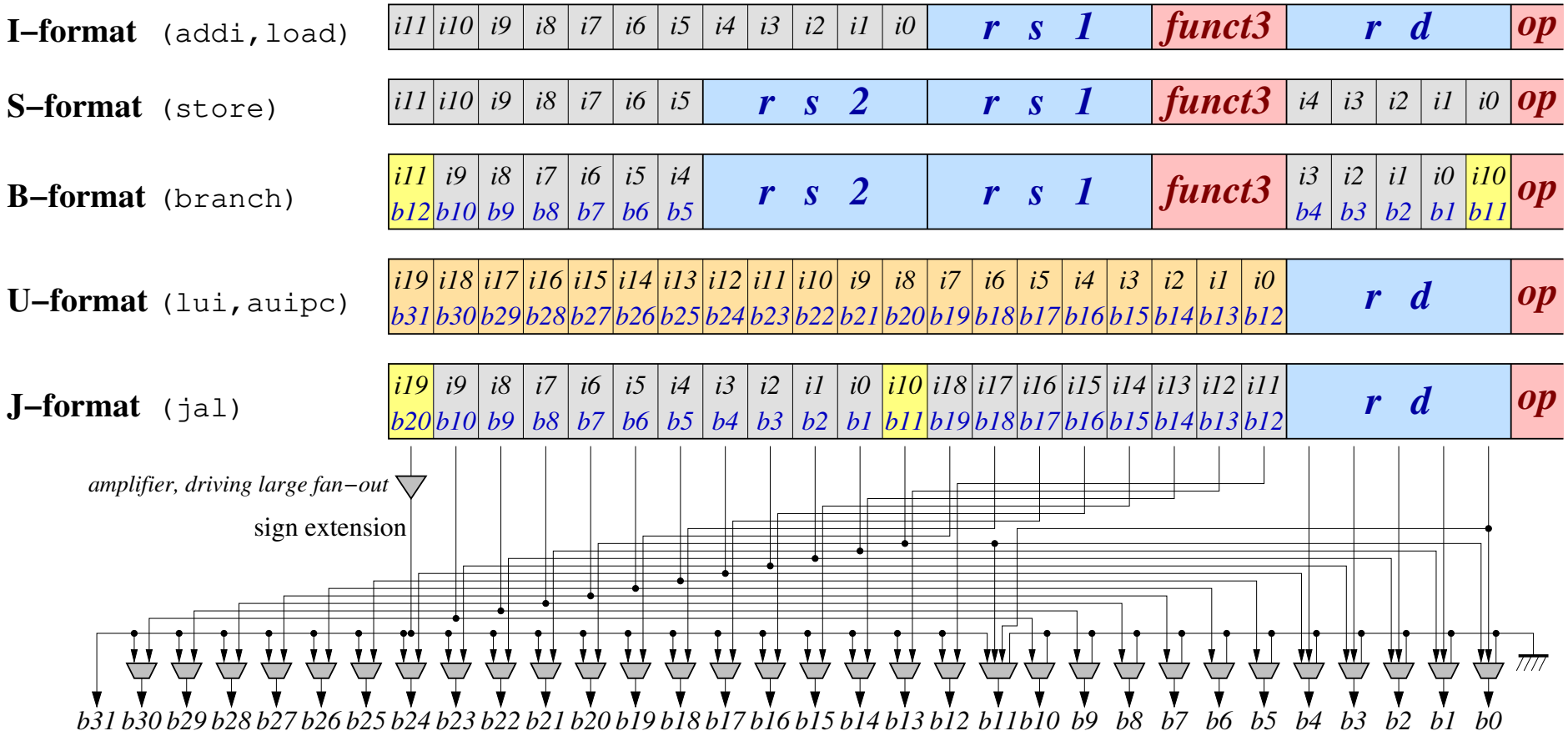
- κι ακόμα παραπάνω, διότι εδώ λείπει το U-format

Πού πραγματικά βρίσκονται τα bits των Immediates

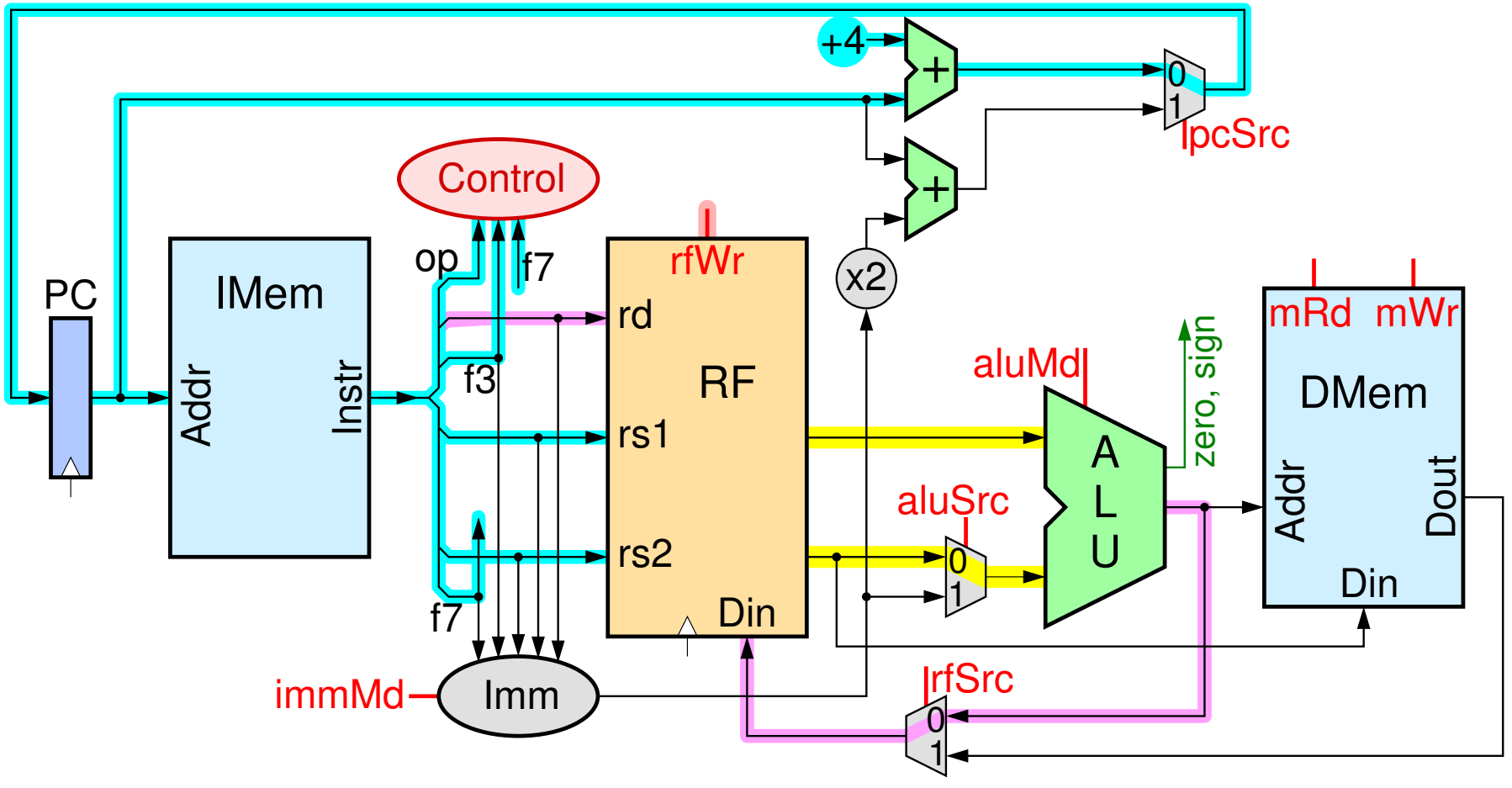


«Μην αναβάλετε γιά Run time ό,τι μπορείτε να κάνετε σε Compile time»

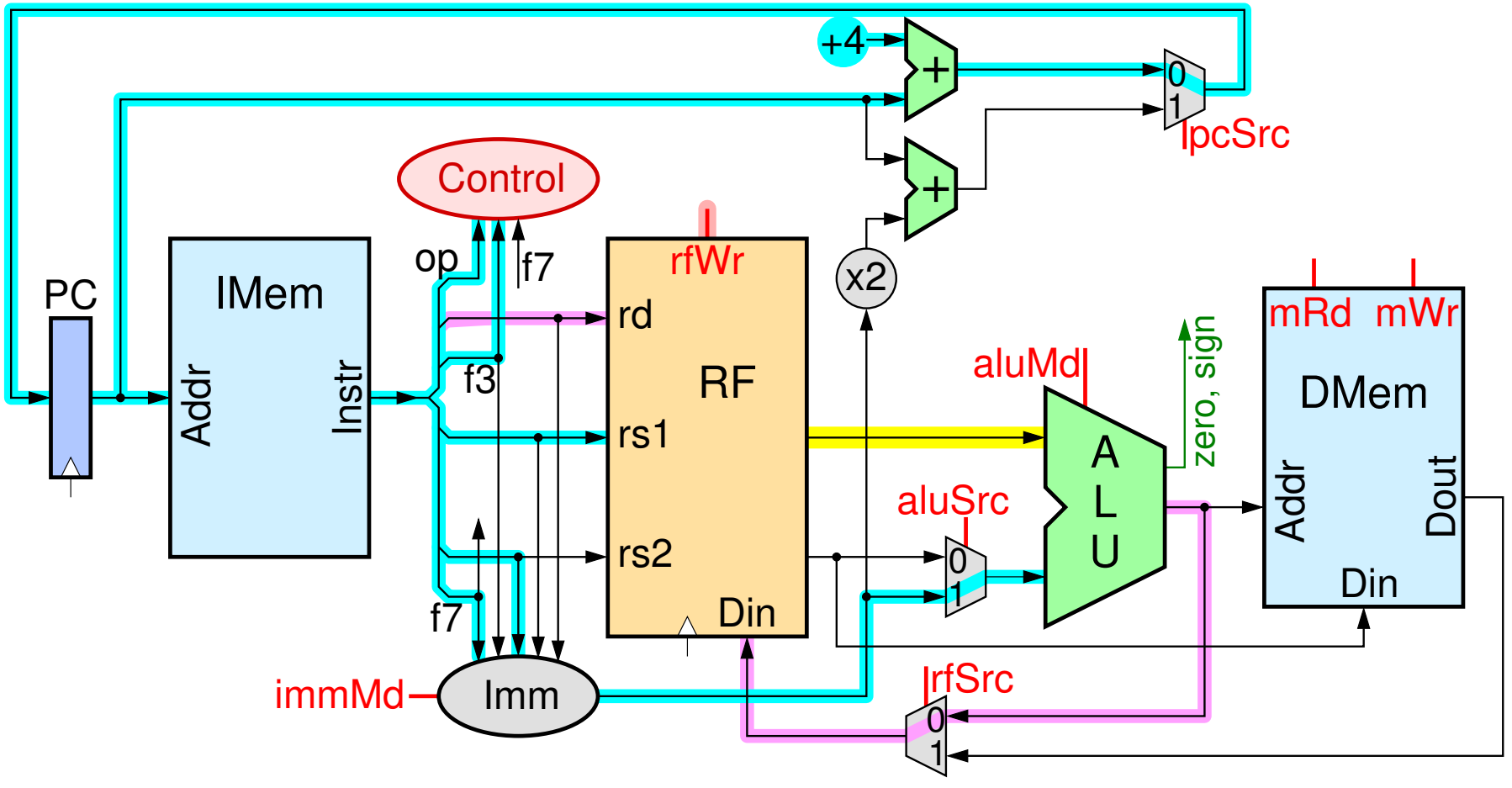
Πολύ λιγότεροι και μικρότεροι πολυπλέκτες έτσι...



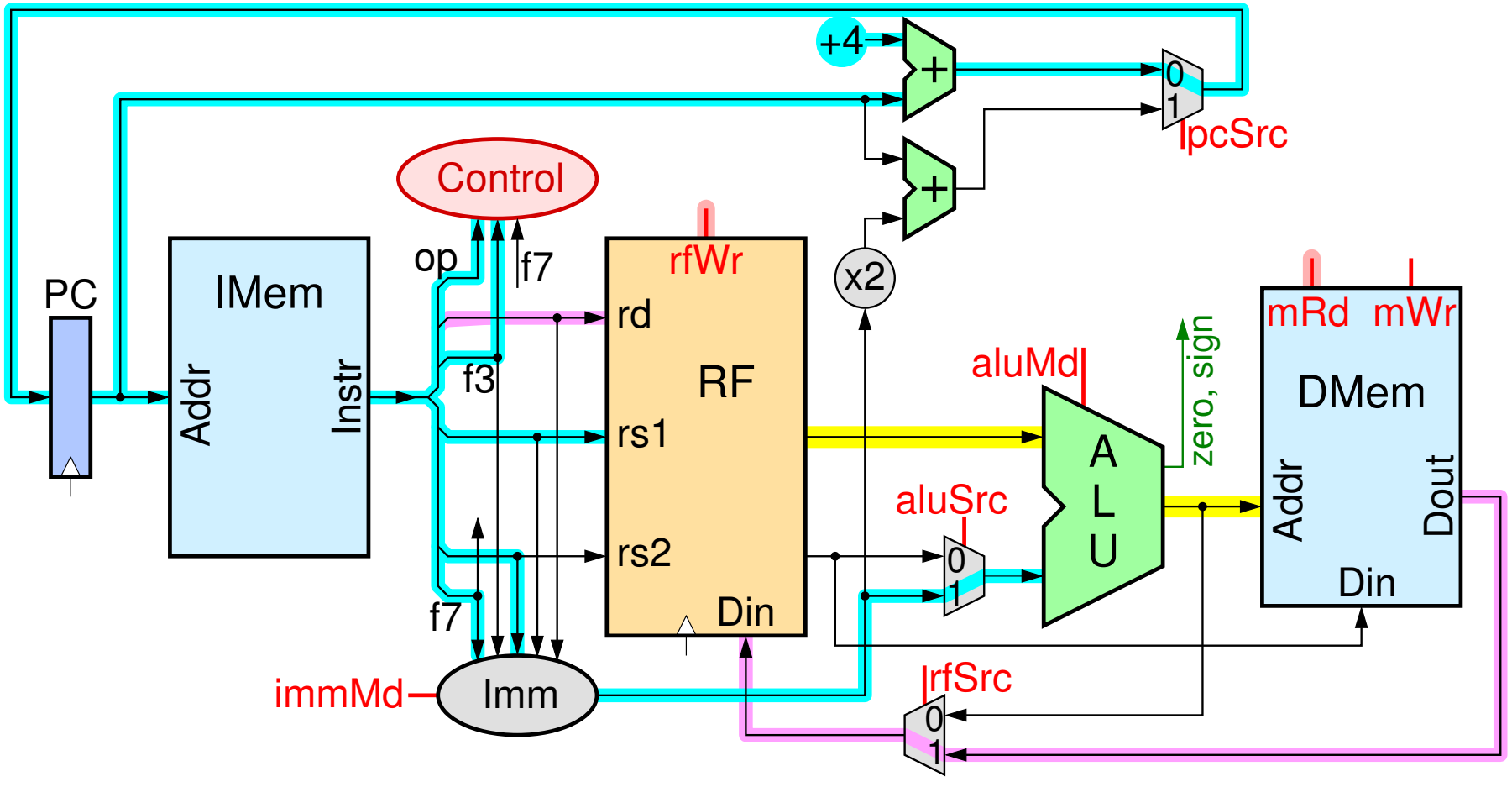
Εντολές *πράξεων* τύπου-**R** (π.χ. add, sub, mul)



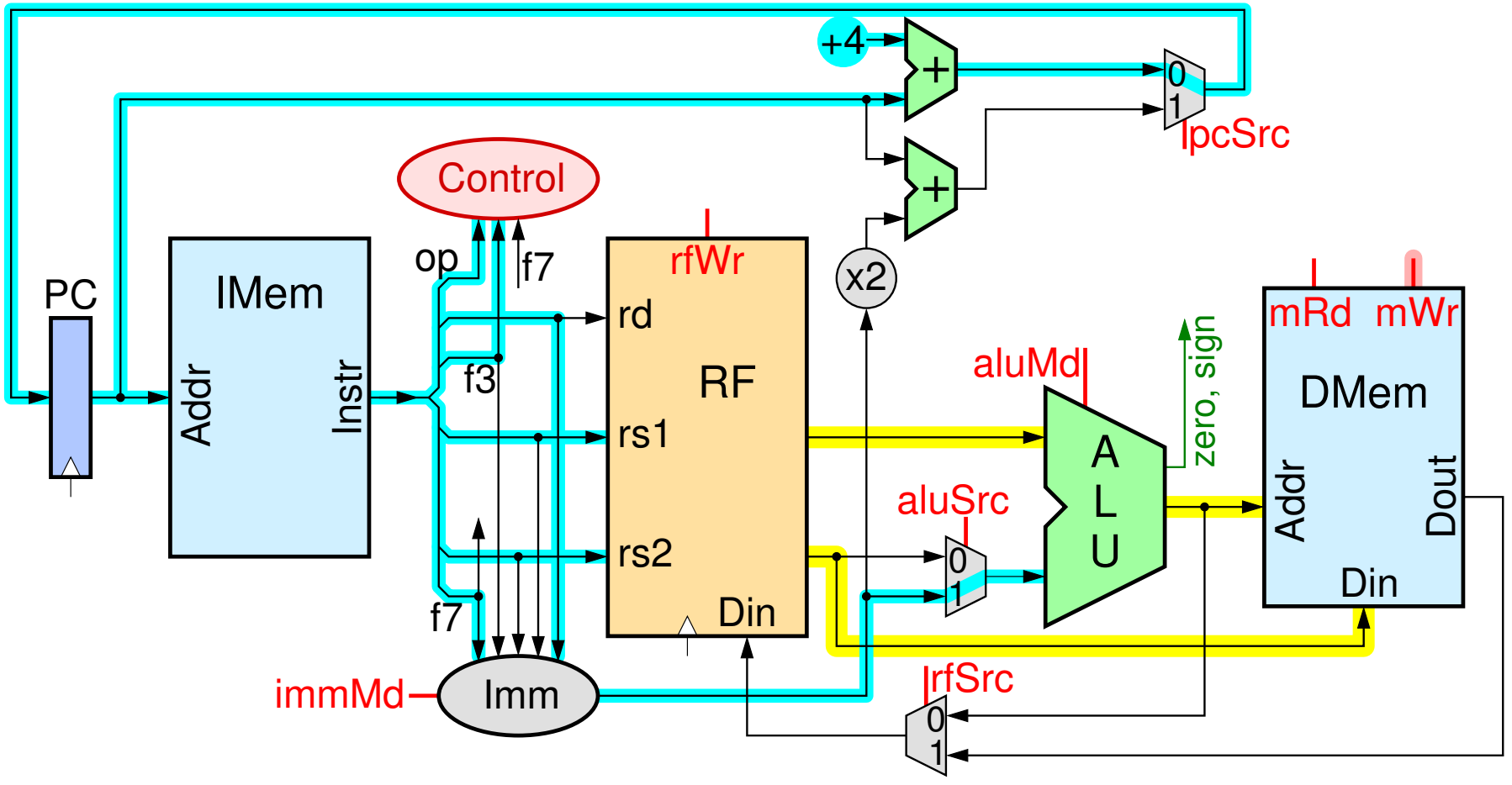
Εντολές *πράξεων* τύπου-**I** (π.χ. addi, slti, xori)



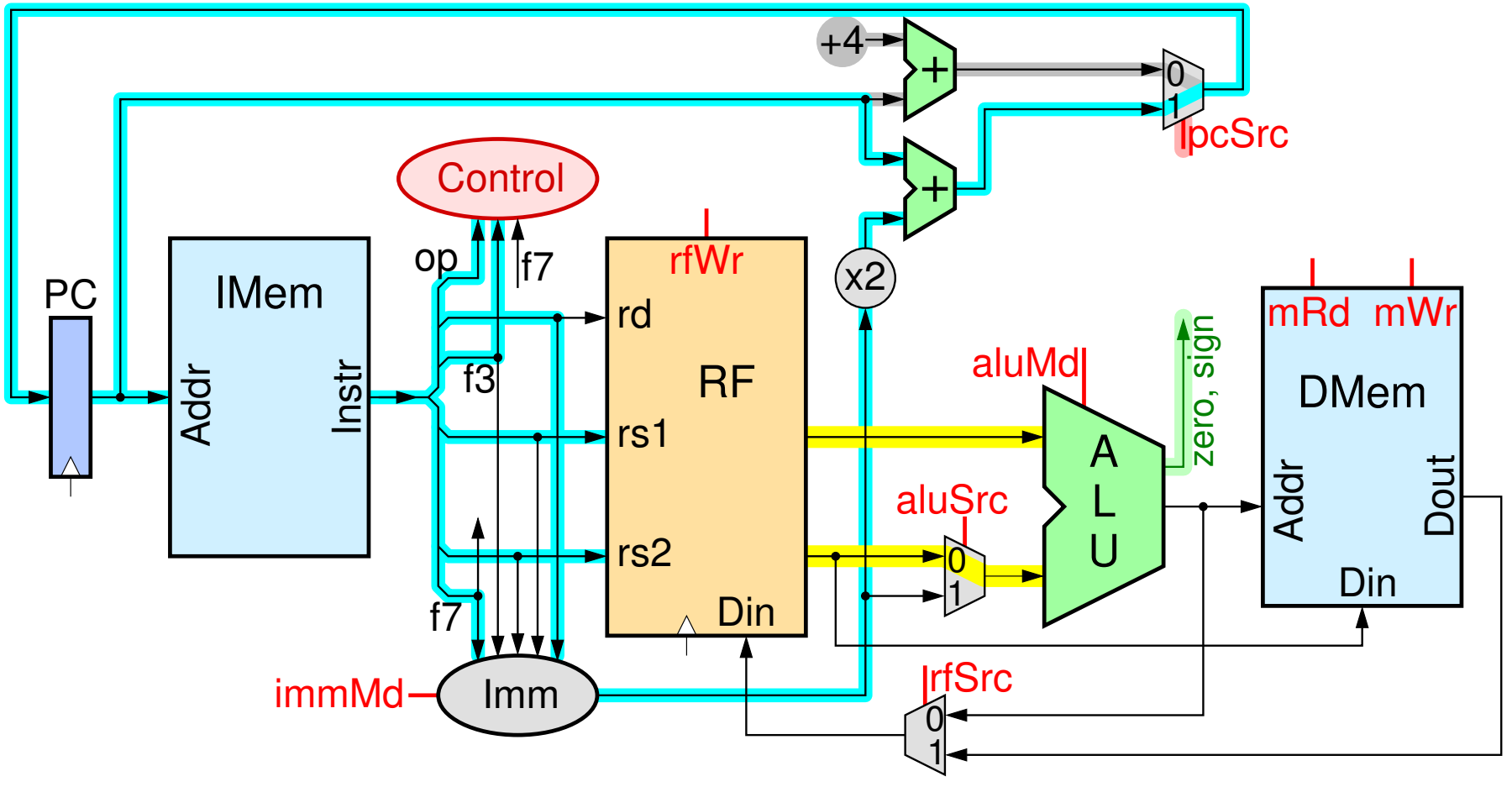
Εντολές *Load* (τύπου-**I**)



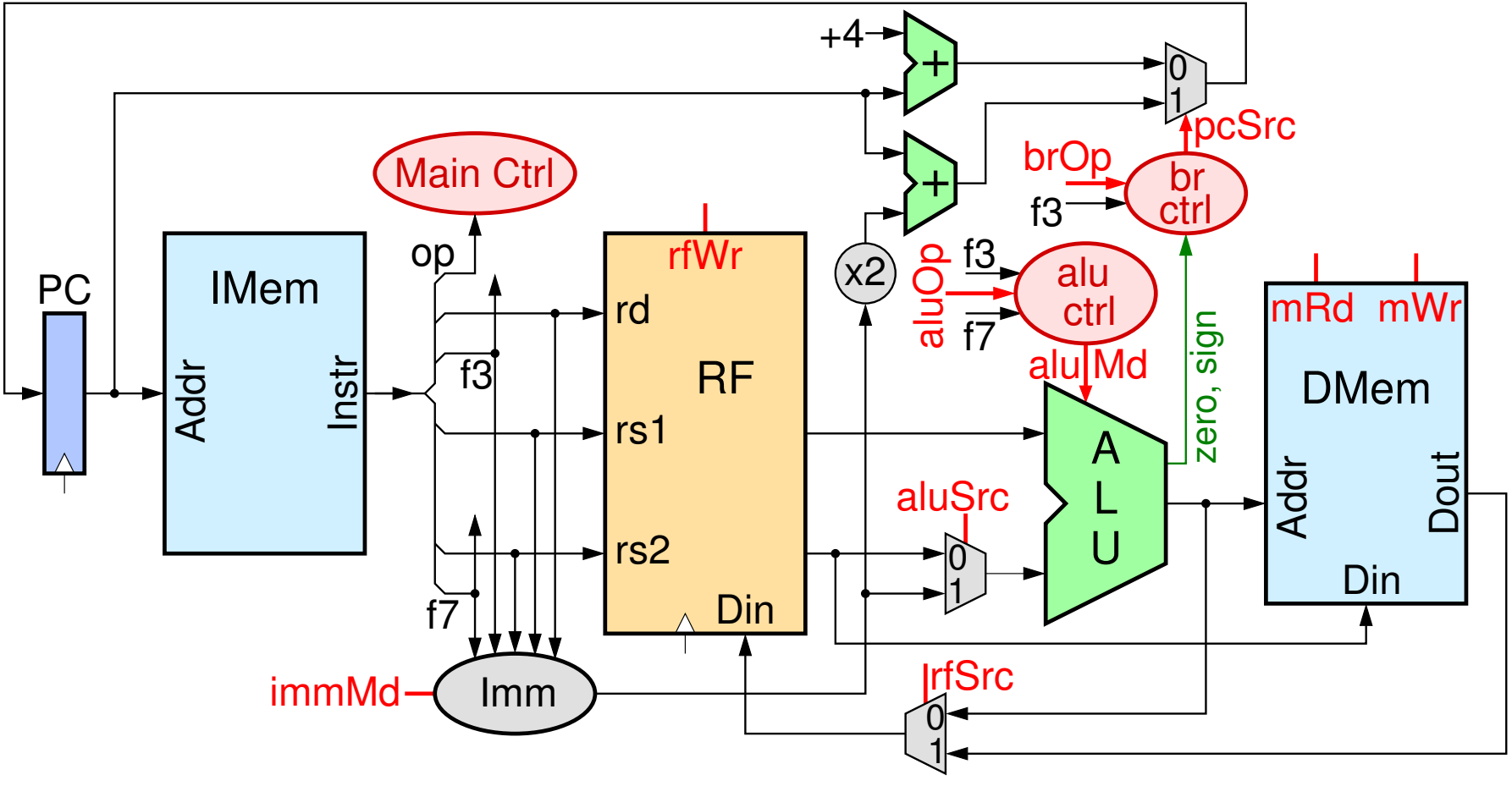
Εντολές *Store* (τύπου-**S**)



Εντολές *Branch* (τύπου-**B**)



Το (συνδυαστικό) Κύκλωμα Ελέγχου – Άσκηση



Ο βασικός Opcode του RISC-V

floating-point

memory fence,
Atomic Memory Operations

32-μπιτες πράξεις
στον RV64

inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 (> 32b)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48b
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	64b
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	48b
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	≥ 80b

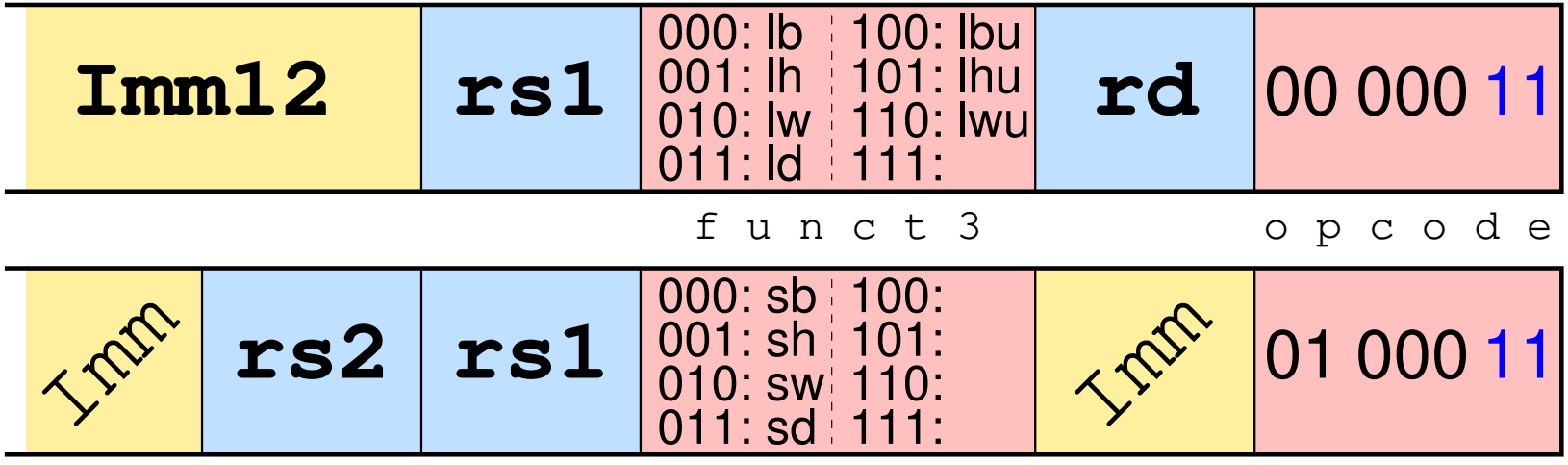
Table 24.1: RISC-V base opcode map, inst[1:0]=11

fused multiply-add
(floating-point)

ecall, CSR rd/wr
(Control Status Registers)

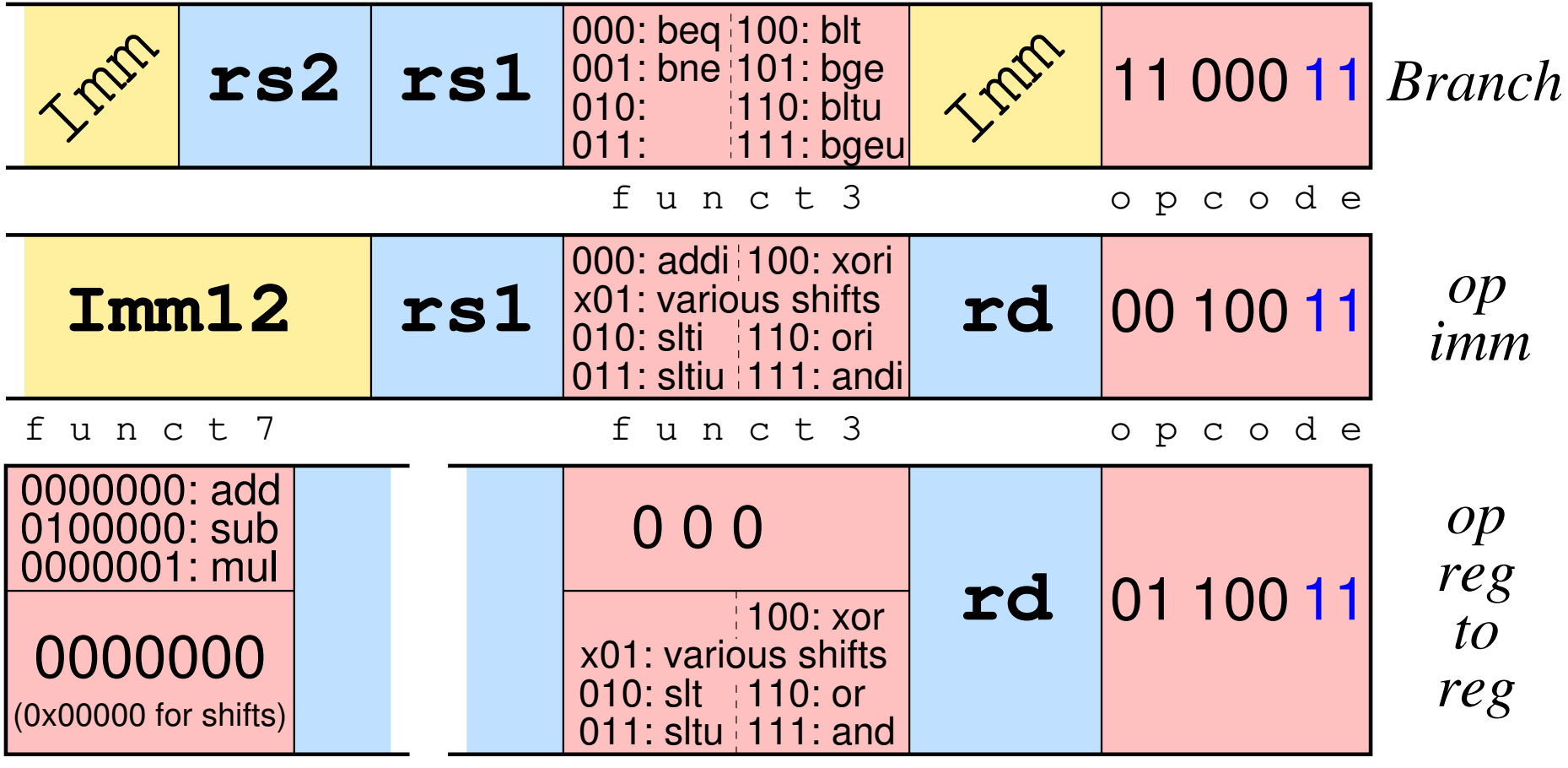
- Βασικός Opcode: 7 bits
- 2 LS bits = 11 γιά 32-μπιτες εντολ.
- 5 MS bits ⟶ 32 συνδυασμοί, εδώ
- Reserved = δεσμ. γιά μελλ. χρ.
  - rv128: γιά μελλοντικό 128-μπιτο
- Custom = γιά ιδιωτική χρήση

Εντολές *Load* και *Store*: κώδικες `funct3`

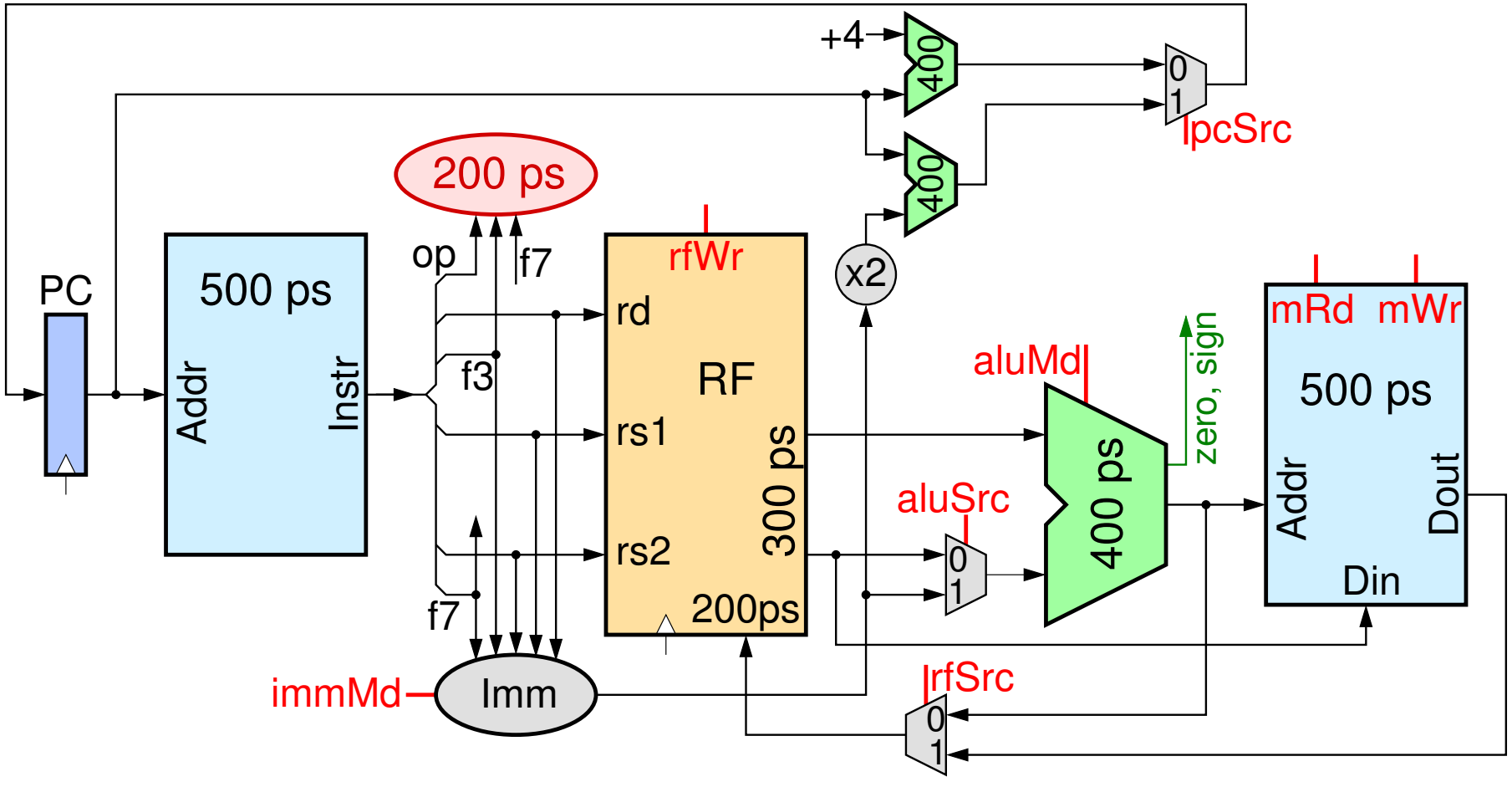


- Ομοιομορφία Opcode: διαφέρουν σε 1 bit μόνον
- Ομοιομορφία funct3 (ίδια μεταξύ load και store):
  - MS bit: signed/unsigned
  - LS bits: size
- Διαφορά σε 1 bit μόνο ⇒ γειτονικά τετράγ. σε χάρτη Karnaugh

Εντ. Διακλάδωσης και Πράξεων: κώδικες funct3/7



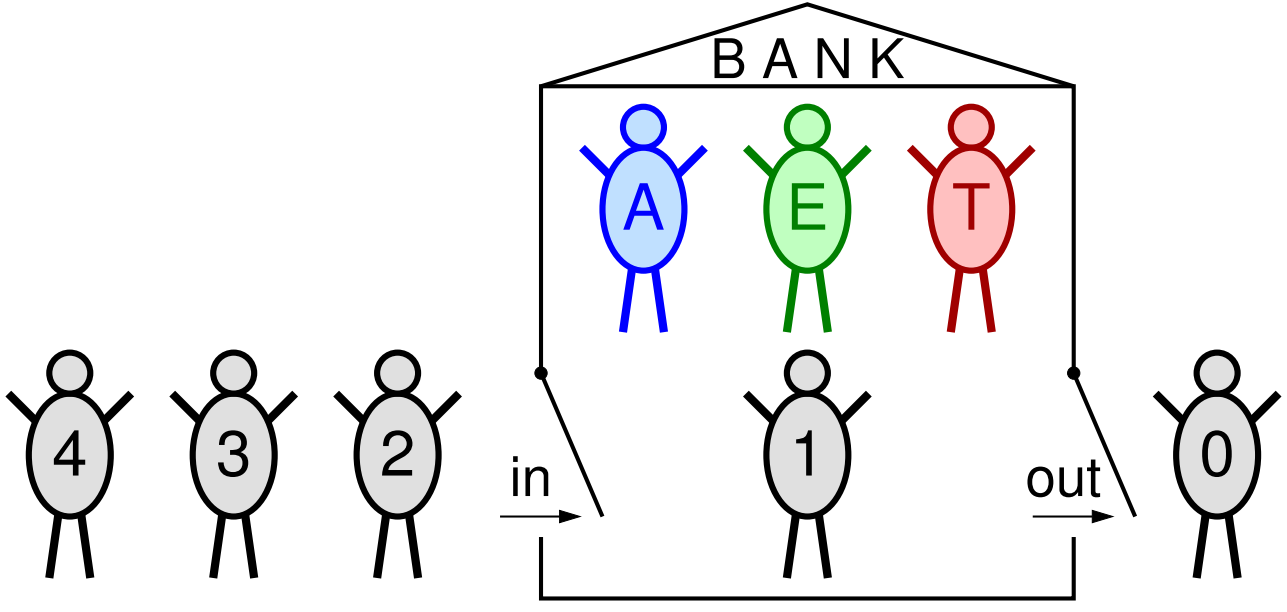
Ποιός εργάζεται πότε; Πόσο αργό το Ρολόϊ; – Άσκηση



Ποιός εργάζεται πότε; Περί Σπατάλης Πόρων

- Ειδικευμένοι Πόροι
  - μόνον ένας σε θέση να κάνει την κάθε εργασία
- Σειριακή Εξάρτηση
  - η μία εργασία πρέπει να τελειώσει πριν αρχίσει η επόμενη
- Ως Γραμμή Παραγωγής
  - Ομοχειρία; (*Pipelining*)

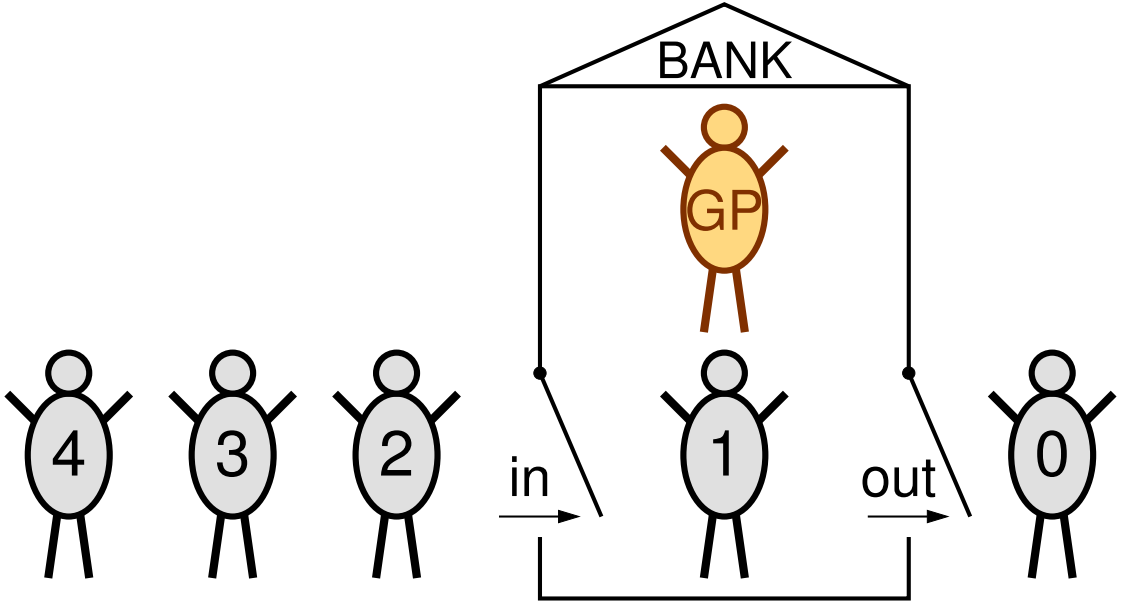
Γραμμής Παραγωγής με ειδικευμένους υποαπασχολ.



- Ένας μόνον πελάτης εντός ανά πάσα στιγμή (!)
- Παροχή = 1 πελ. / 30 λ.
- Απασχόληση προσωπικού = 1/3
- Γιατί τρείς υπάλληλοι;;;

- A: συμπλ. Αίτησης Δανείου (10 λεπτά)
- E: Έλεγχος/έγκριση αίτησης (10 λεπτά)
- T: Ταμείο-εκταμίευση δαν. (10 λεπτά)

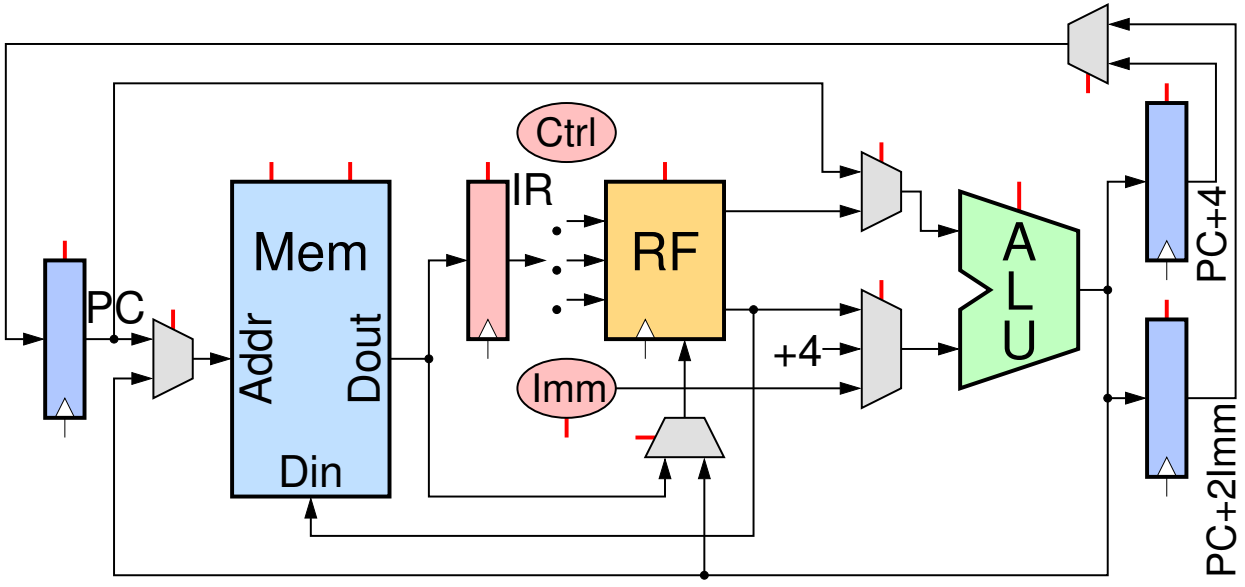
Εξοικονόμηση πόρων με Πόρους *Γενικού Σκοπού*



- Ένας μόνον πελάτης εντός ανά πάσα στιγμή
- Παροχή = 1 πελ. / 30 λ.
- Απασχόληση προσωπικού = 100%

- GP: *"General Purpose"* (γενικού σκοπού) ξέρει να κάνει όλες τις εργασίες (χρόνος εξυπηρέτησης 30 λεπτά ανά πελάτη, πάλι)

Παραλλαγή του υπολογιστή με εξοικονόμηση πόρων



- Μία ALU μόνον, αντί ALU + 2 adders πριν
- Μία μνήμη μόνον, αντί δύο μνήμες πρίν
- Περίπου 5 φορές γρηγορότερο ρολόϊ, και
- Πέντε κύκλοι του γιά την χειρότερη εντολή
- Άλλες εντολές σε λιγότερους κύκλους