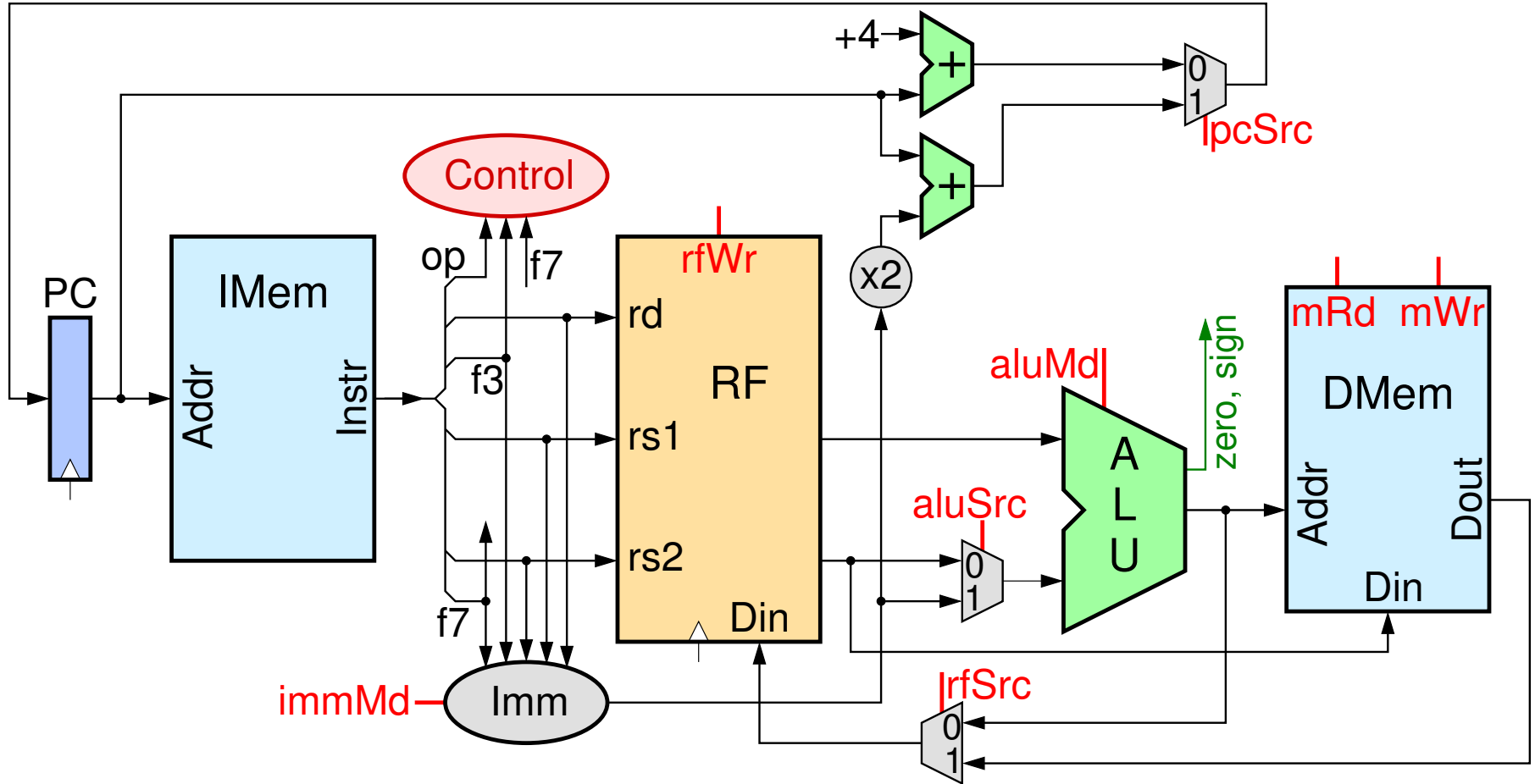


Μία Απλή Υλοποίηση του RISC-V σε έναν (μακρύ) Κύκλο Ρολογιού ανά Εντολή

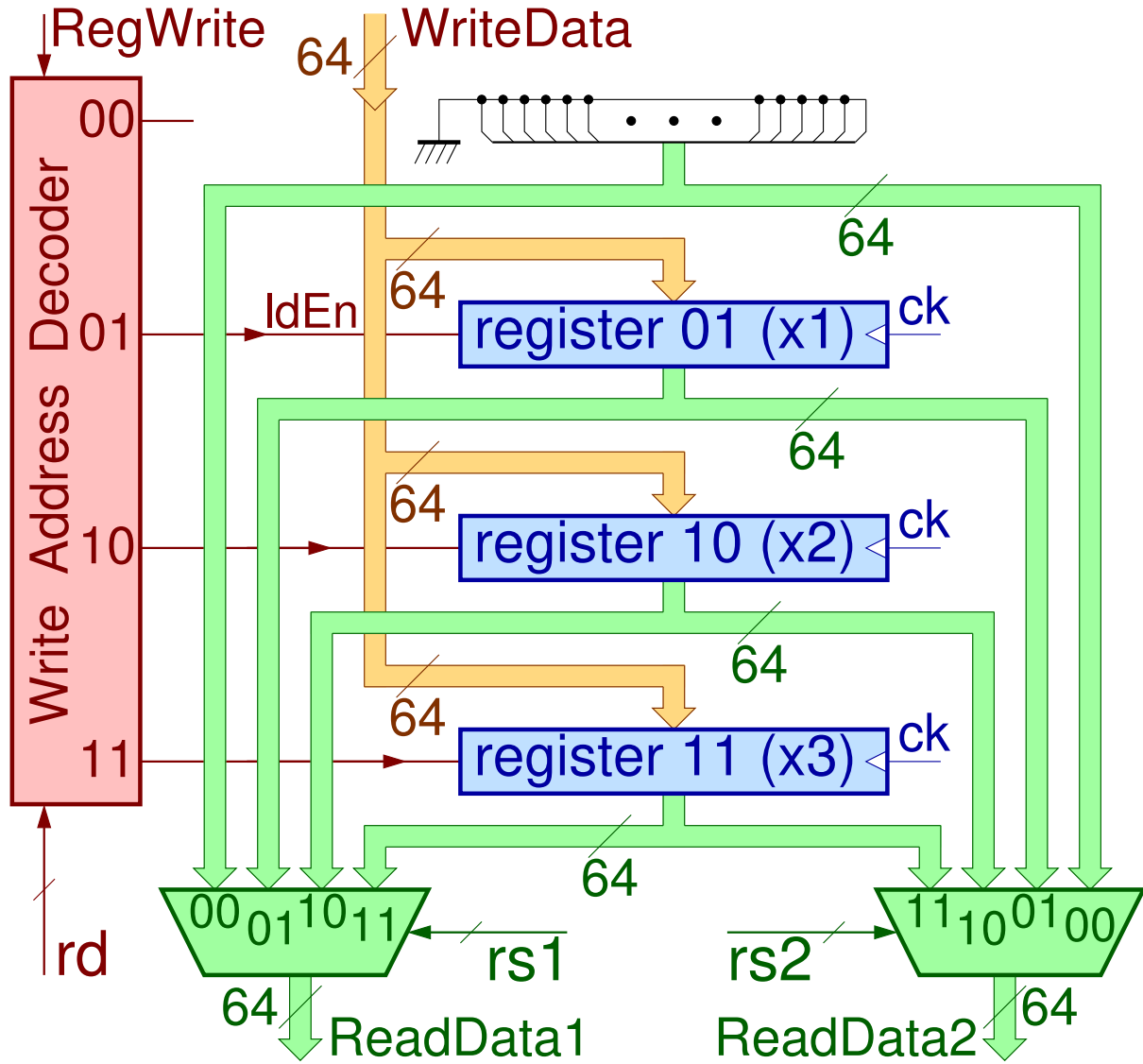
08α (§8.1-8.8) – 10-13 Μαρτίου 2023 – Μανόλης Κατεβαίνης

Διαδρομή Δεδομένων (Datapath) για τις βασικές εντ.

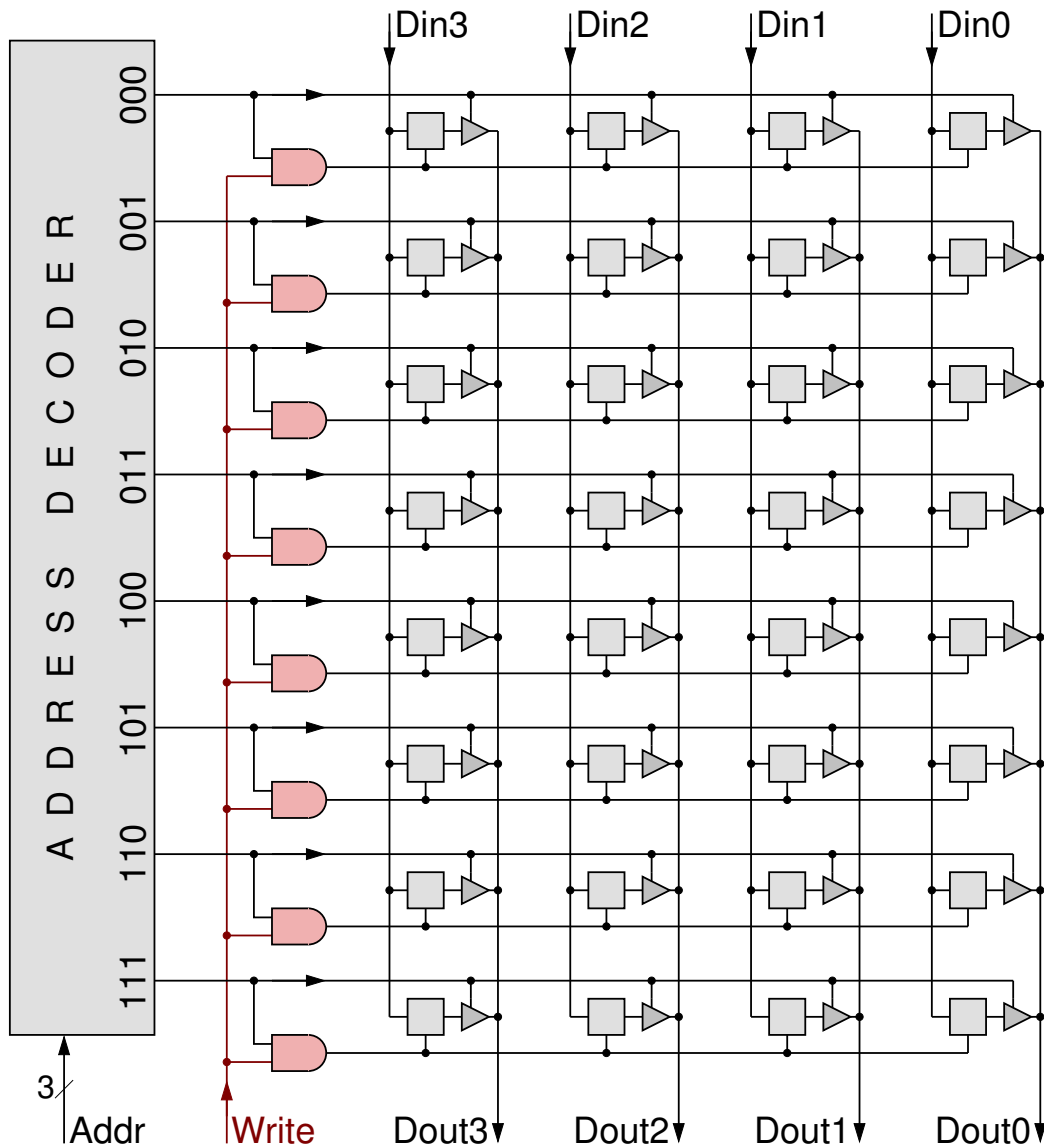


Τρίτο Αρχείο Καταχωρητών

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

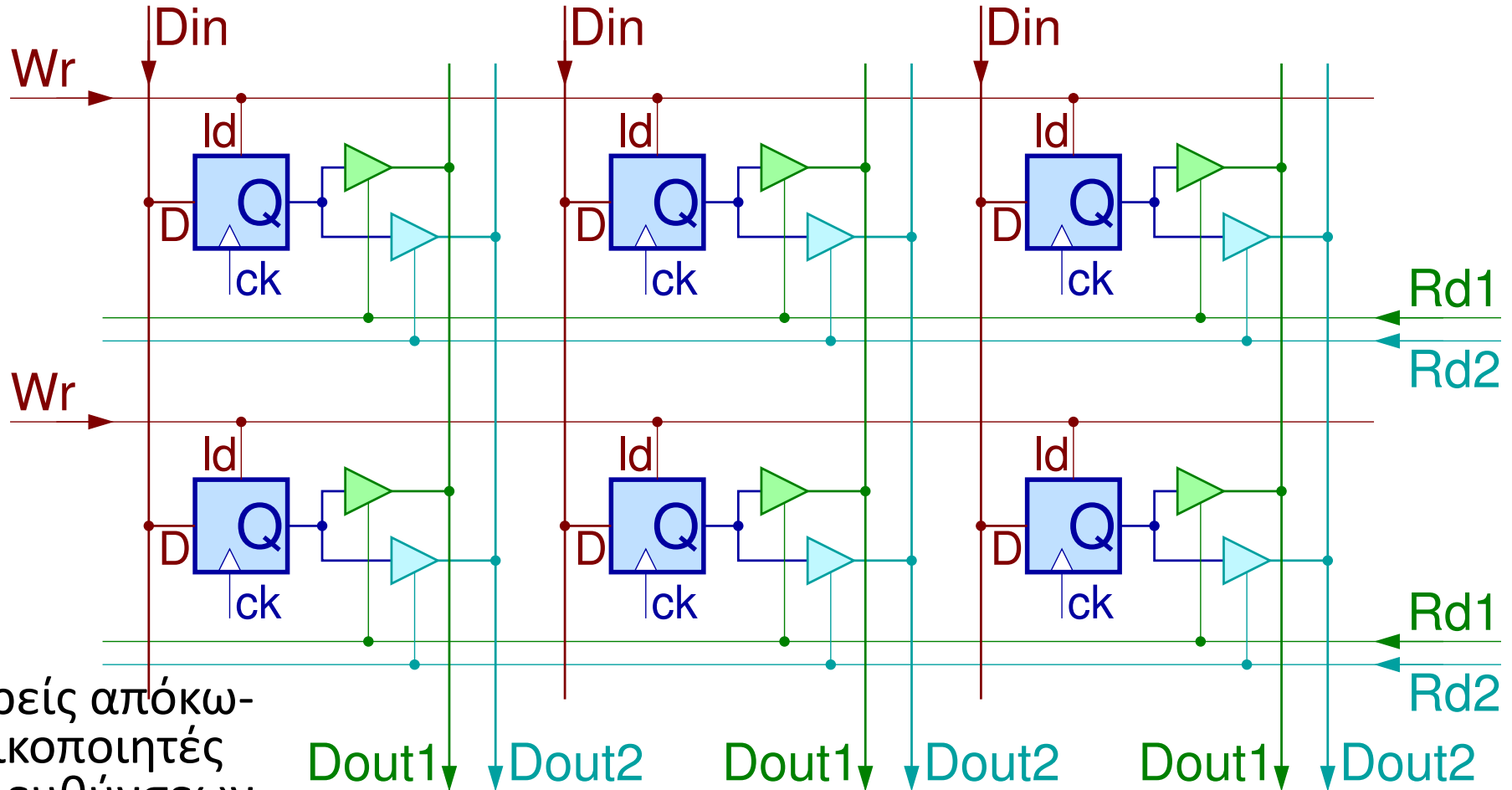


Μνήμη – από ΗΥ-120



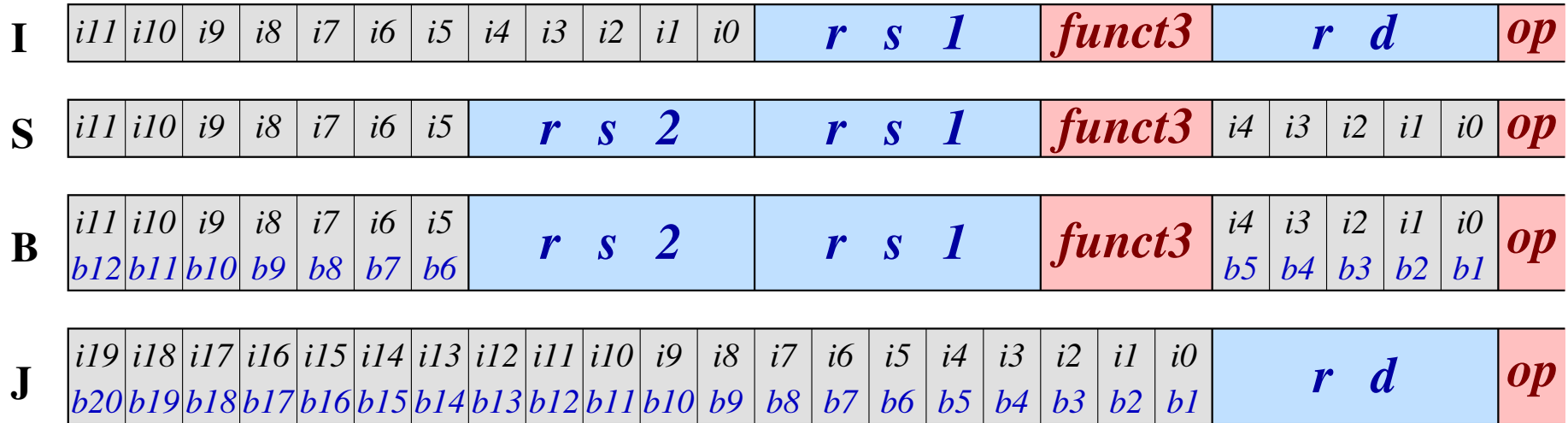
- 8 λέξεις × 4 bits/w
- Μονόπορτη
- Κοινός αποκωδικοπ. διευθύνσεων για αναγνώσεις και εγγραφές – μόνο μία προσπέλαση κάθε φορά – είτε ανάγνωση, είτε εγγραφή
- Επίτρεψη Εγγραφής (Write Enable – WE)

Τρίπορτη Μνήμη (RF): 2x Ανάγνωση & 1x Εγγραφή



Τρεις απόκω-
δικοποιητές
διευθύνσεων

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

I-format (addi, load)



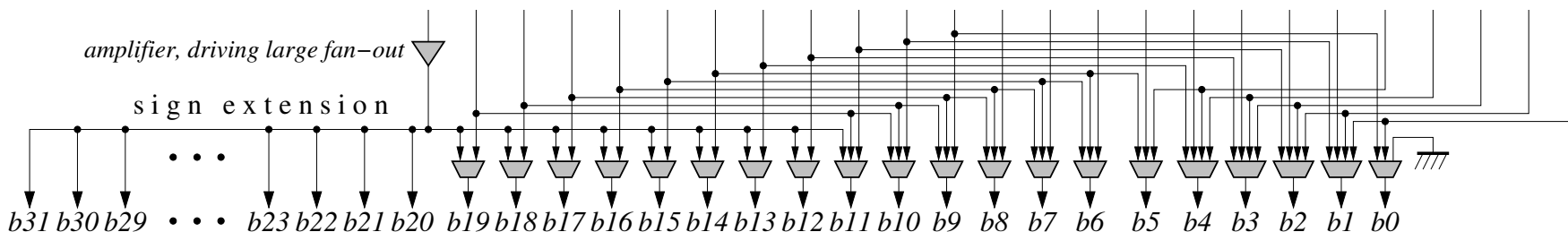
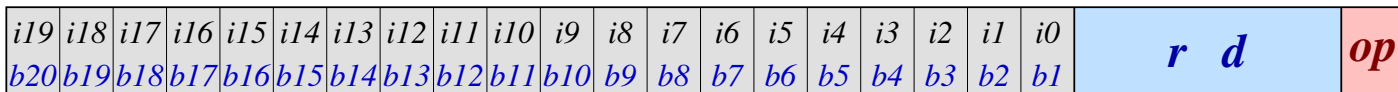
S-format (store)



B-format (branch)

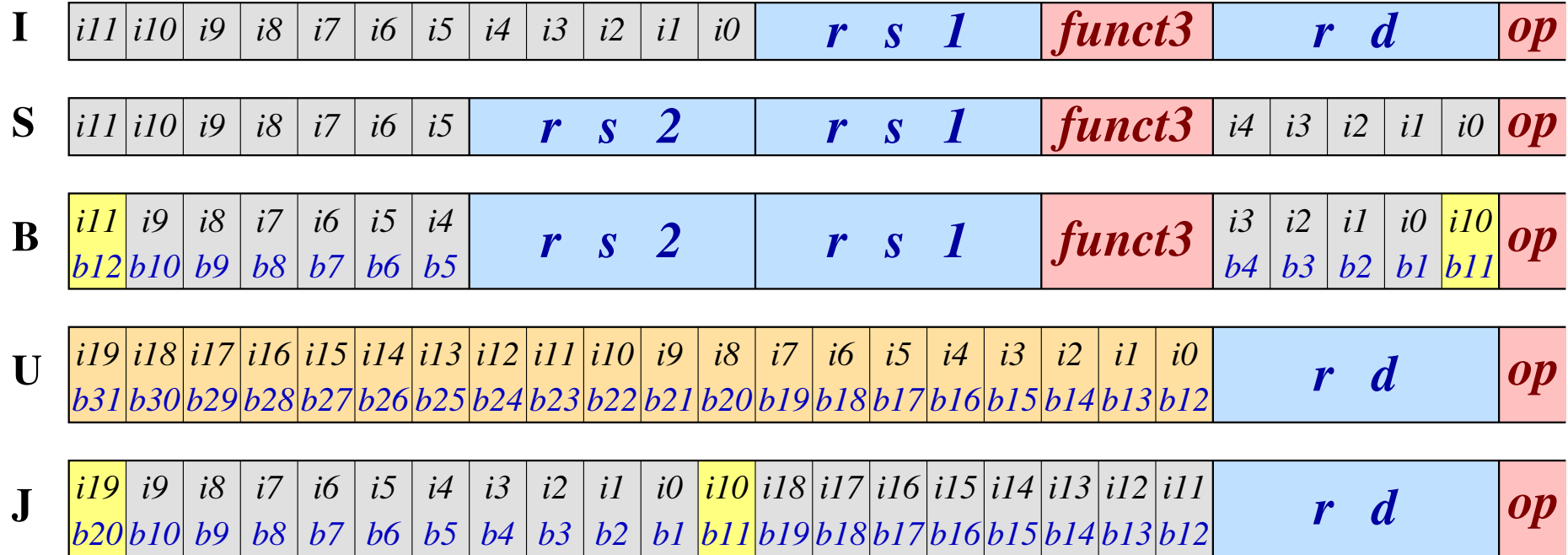


J-format (jal)



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

Πού πραγματικά βρίσκονται τα bits των immediates



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

Πολύ λιγότεροι και μικρότεροι πολυπλέκτες έτσι...

I-format (addi, load)



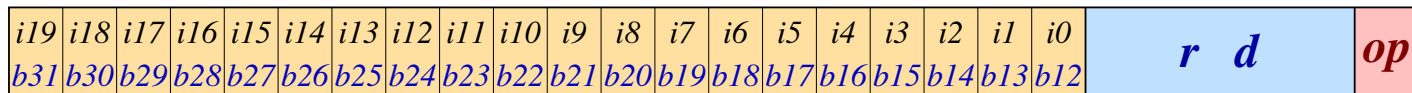
S-format (store)



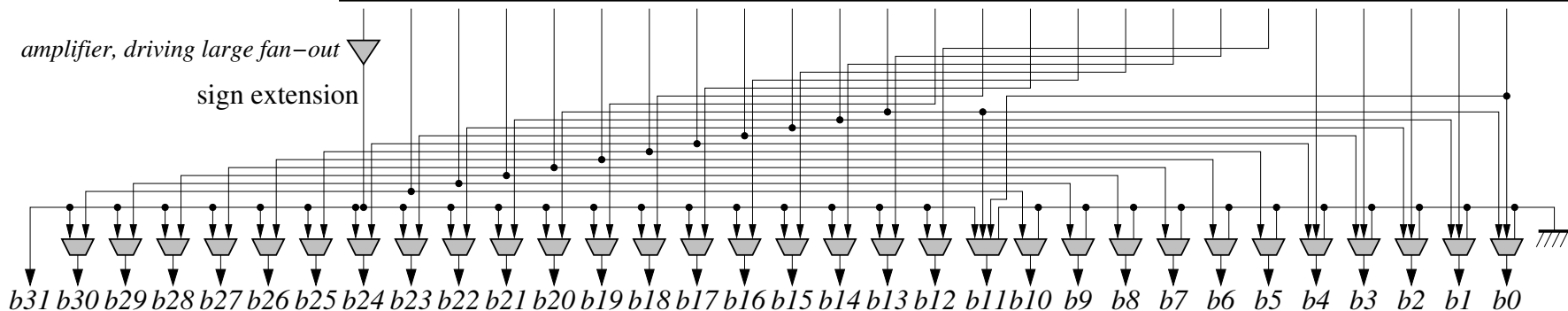
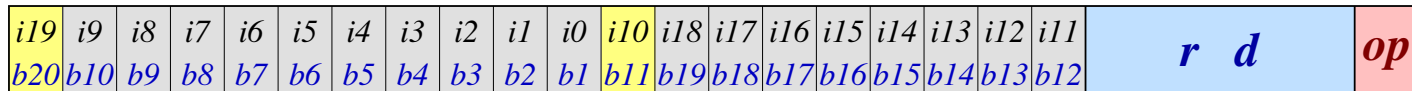
B-format (branch)



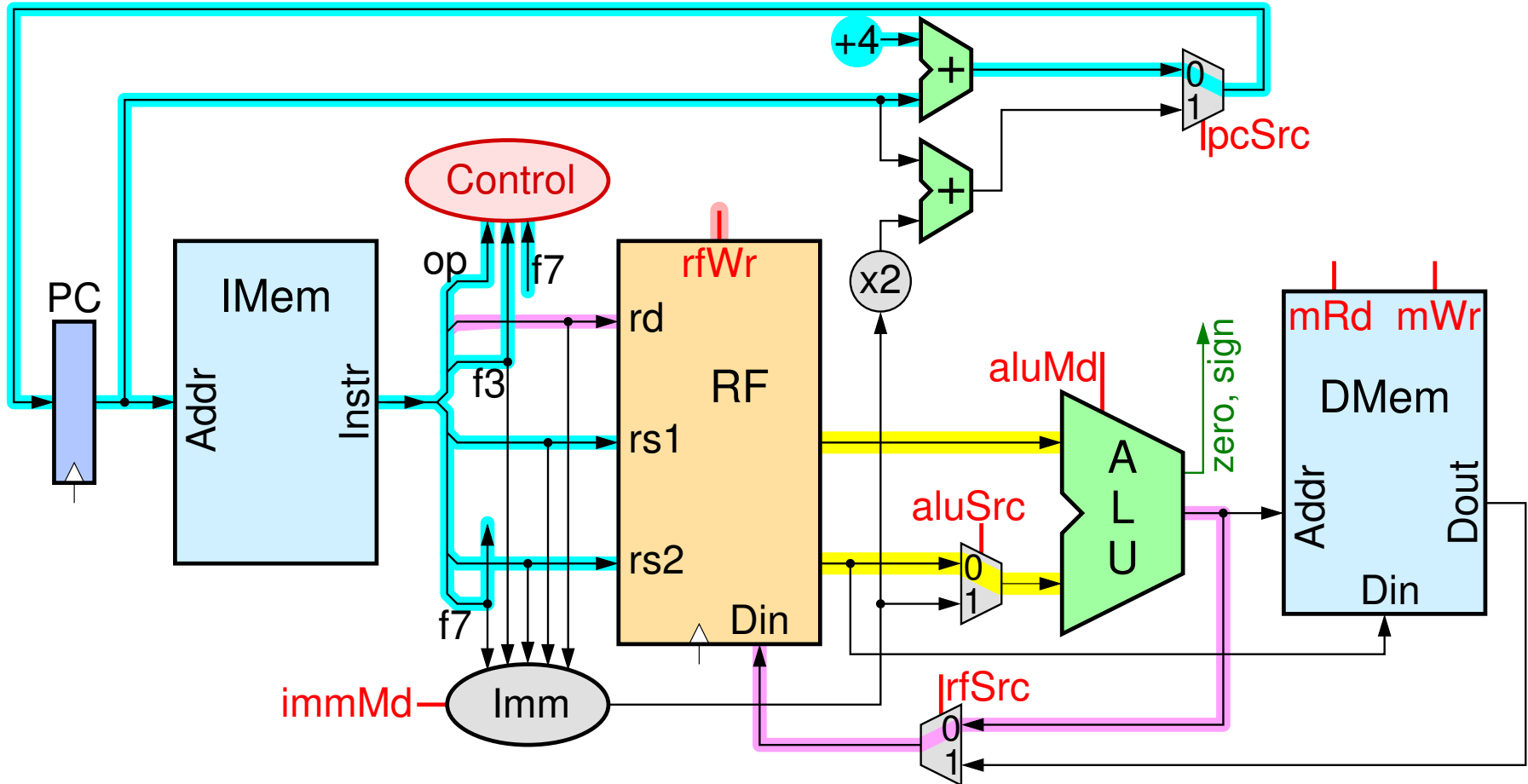
U-format (lui, auipc)



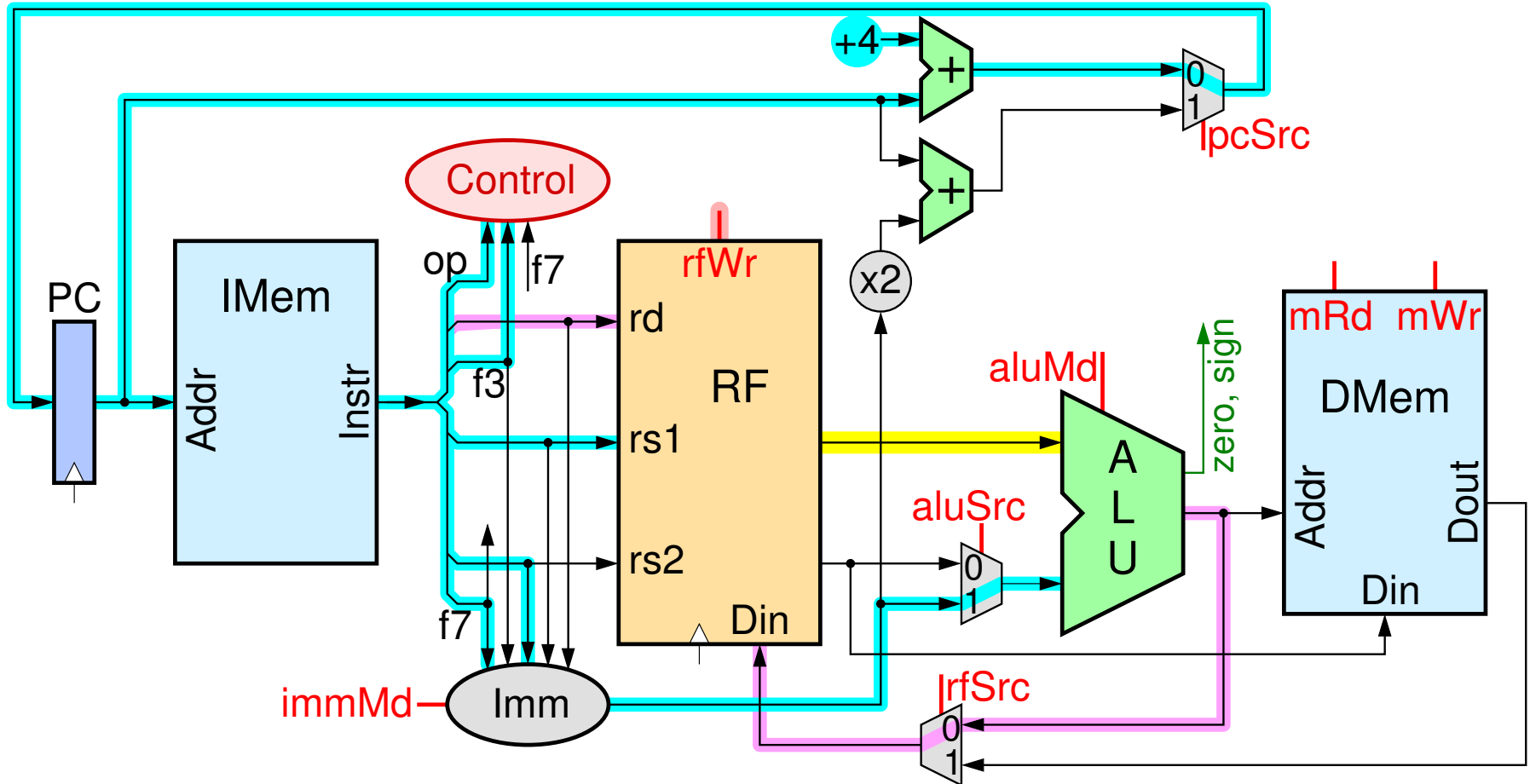
J-format (jal)



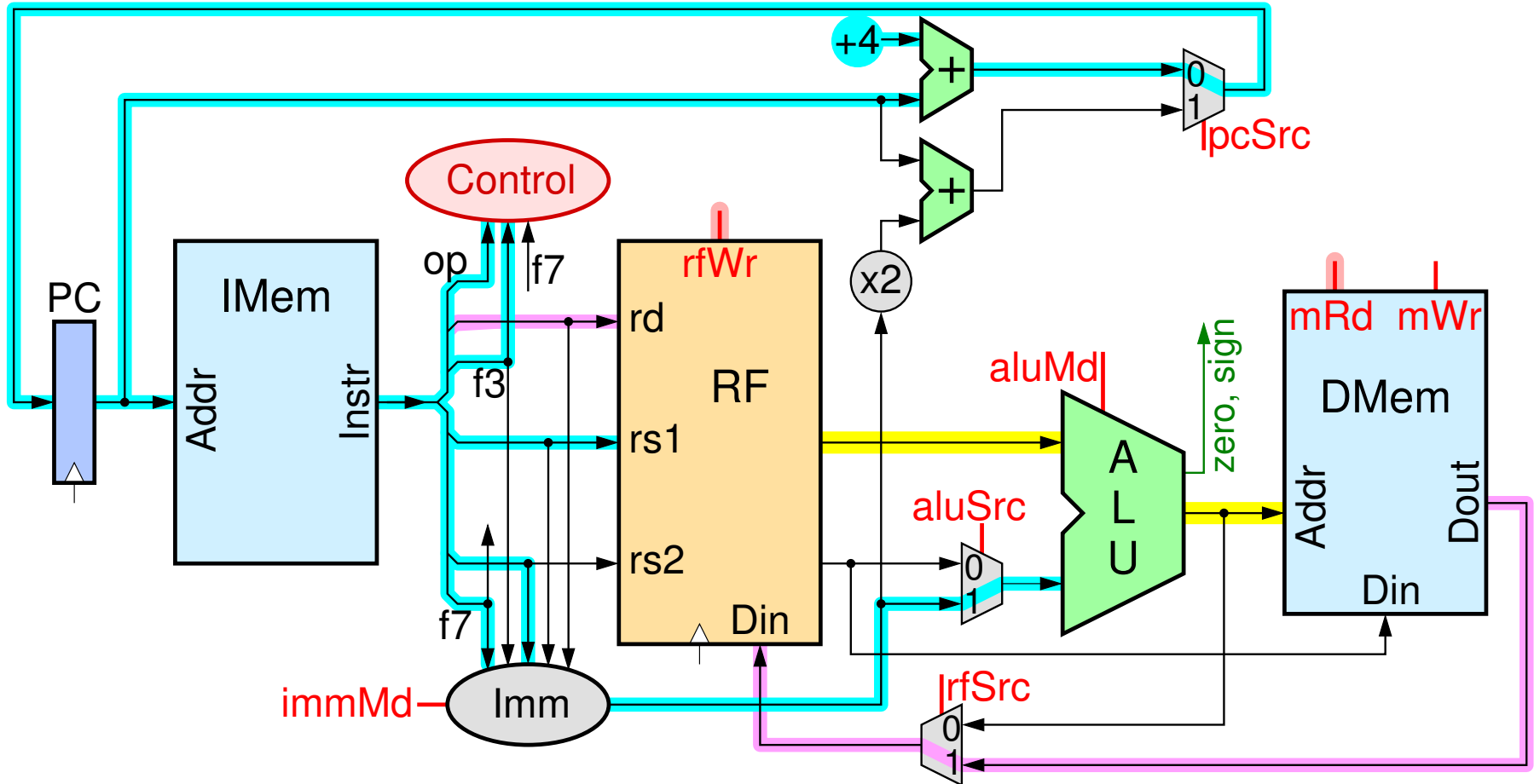
Εντολές πράξεων τύπου-R (π.χ. add, sub, mul)



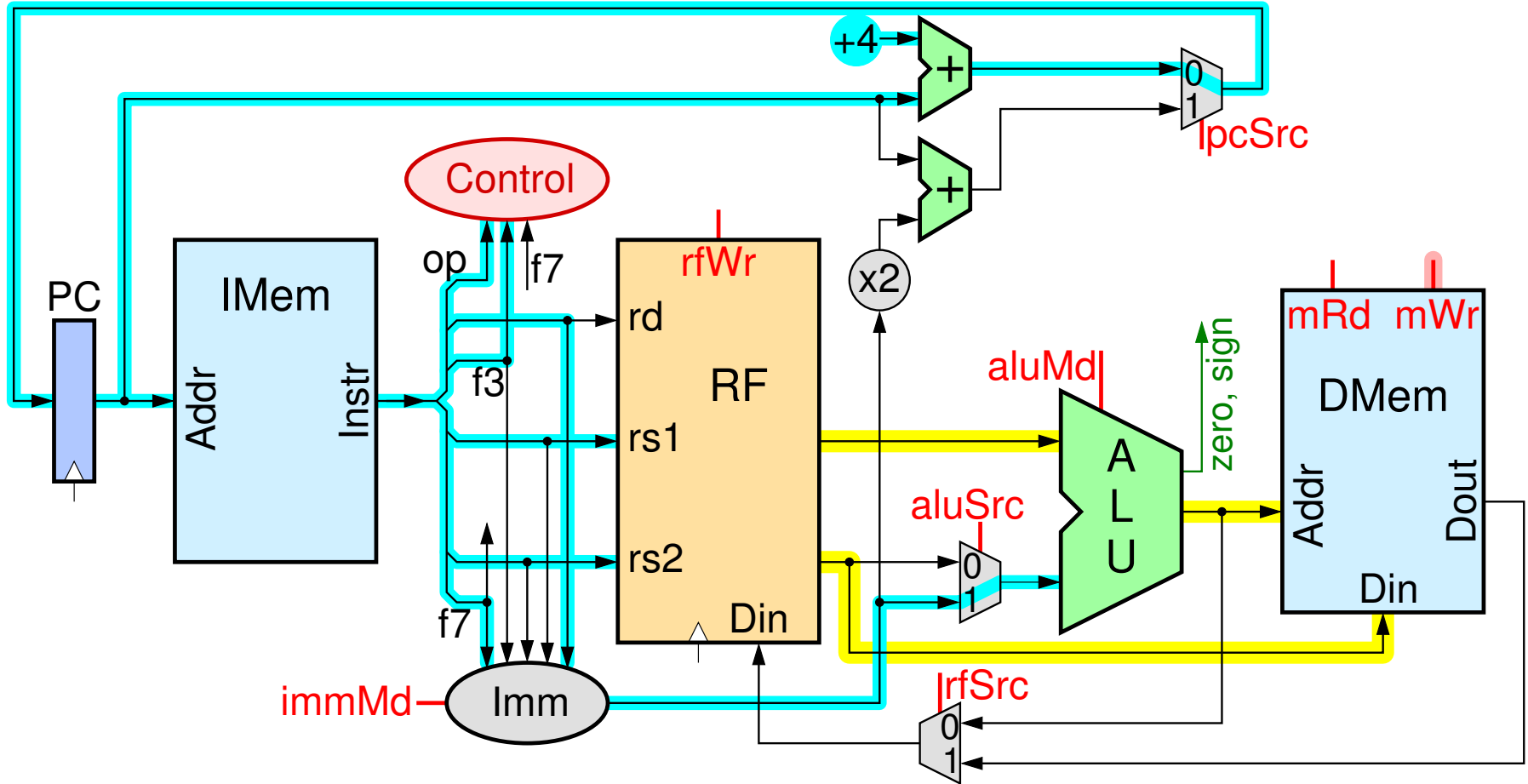
Εντολές πράξεων τύπου-Ι (π.χ. addi, sli, xori)



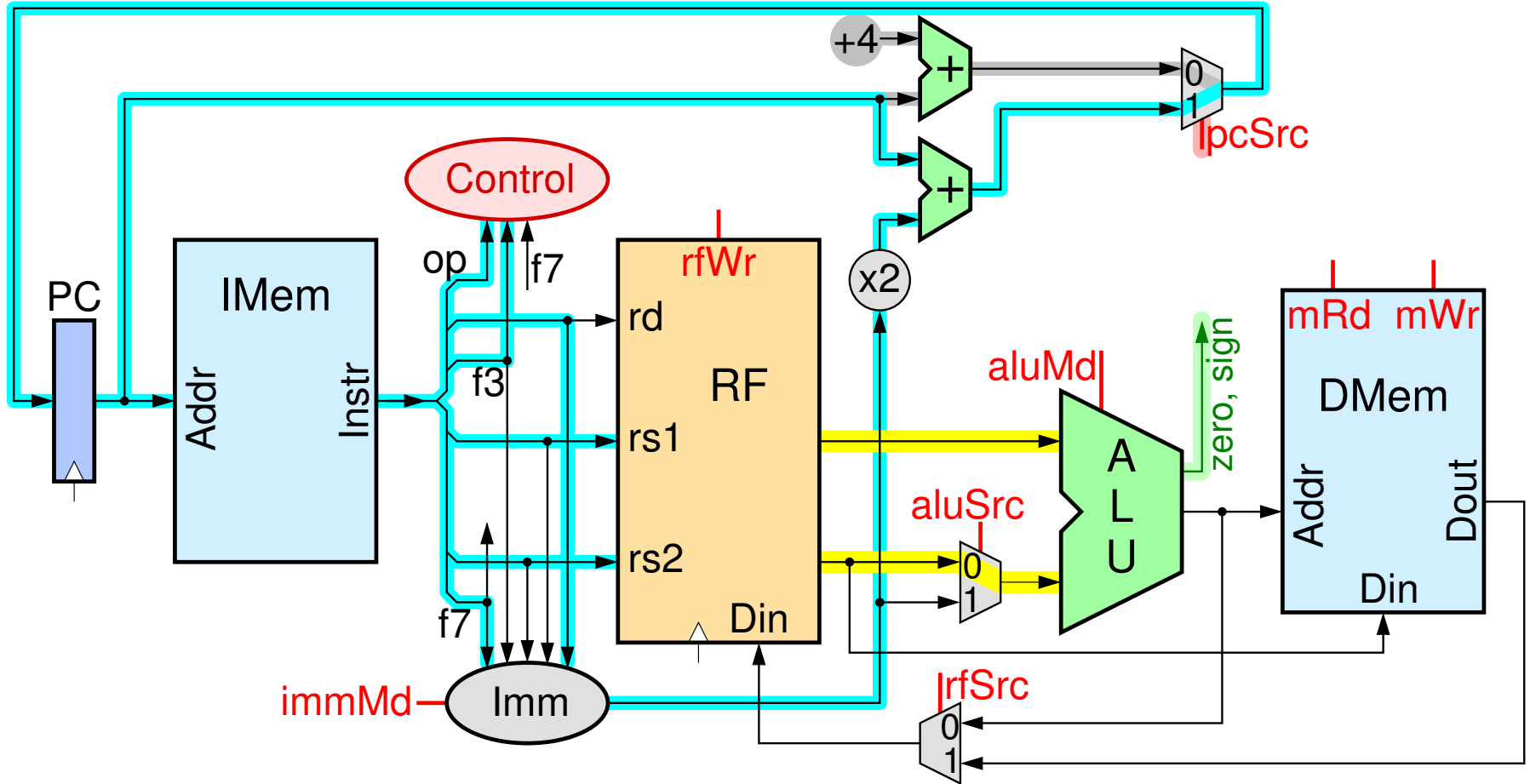
Εντολές Load (τύπου-I)



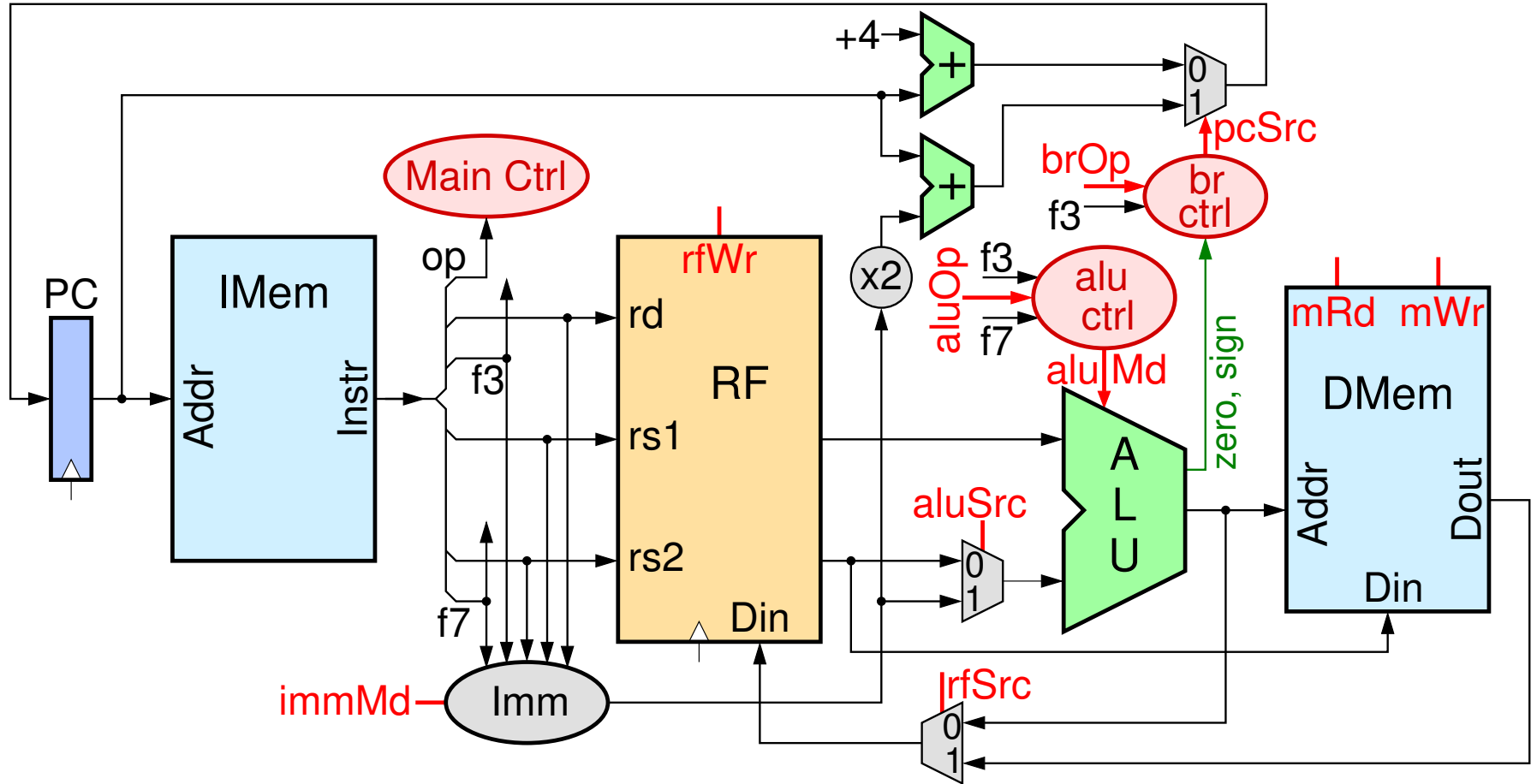
Εντολές *Store* (τύπου-*S*)



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

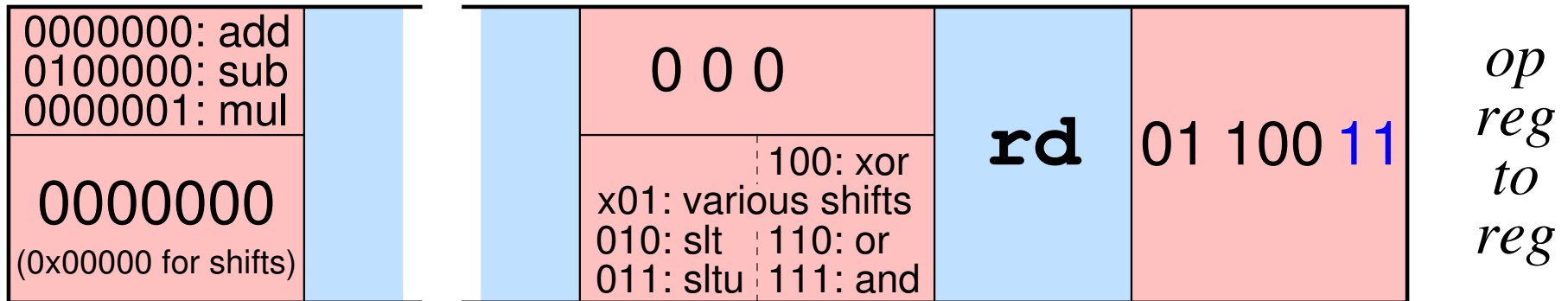
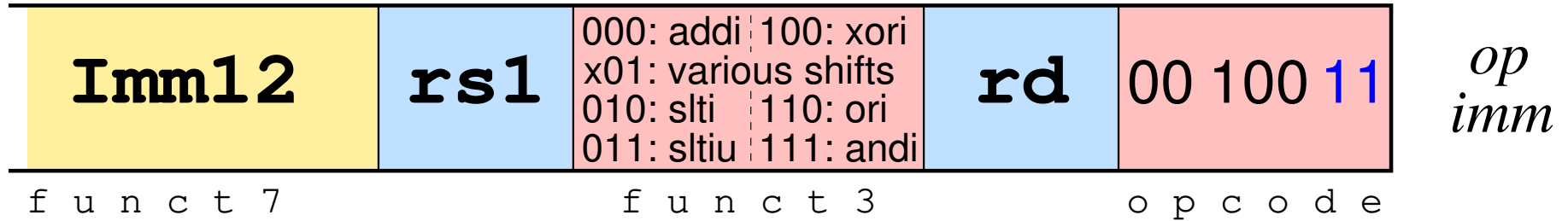
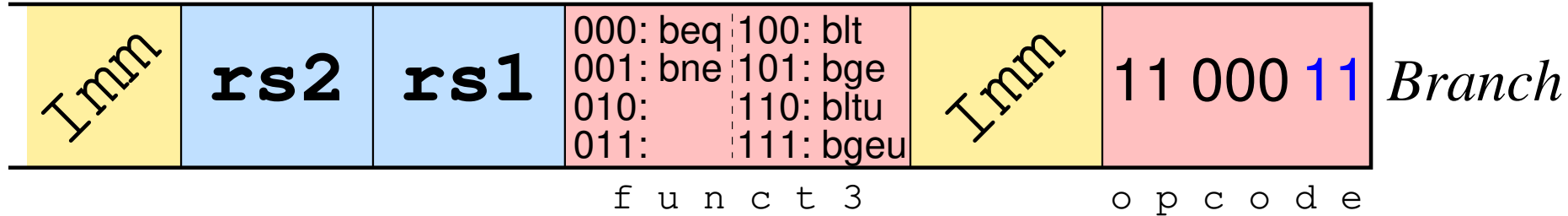
- Βασικός Opcode: 7 bits
- 2 LS bits = 11 για 32-μπιτες εντολ.
- 5 MS bits → 32 συνδυασμοί, εδώ
- Reserved = δεσμ. για μελλ. χρ.
 - rv128: για μελλοντικό 128-μπιτο
- Custom = για ιδιωτική χρήση

Εντολές *Load* και *Store*: κώδικες funct3

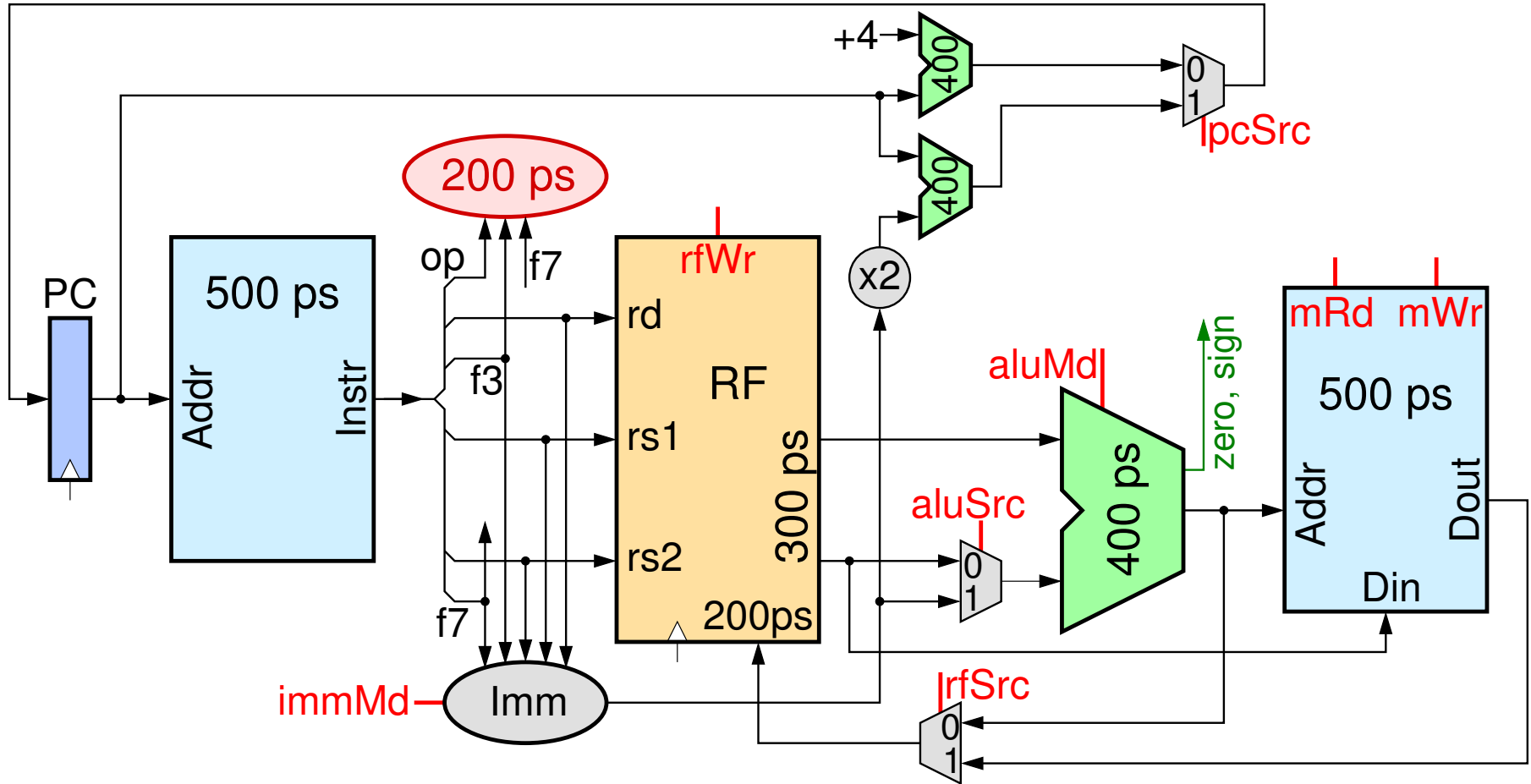
Imm12	rs1	000: lb 100: lbu 001: lh 101: lhu 010: lw 110: lwu 011: ld 111:	rd	00 000 11	<i>Load</i>	
f u n c t 3			o p c o d e			
<i>Imm</i>	rs2	rs1	000: sb 100: 001: sh 101: 010: sw 110: 011: sd 111:	<i>Imm</i>	01 000 11	<i>Store</i>

- 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

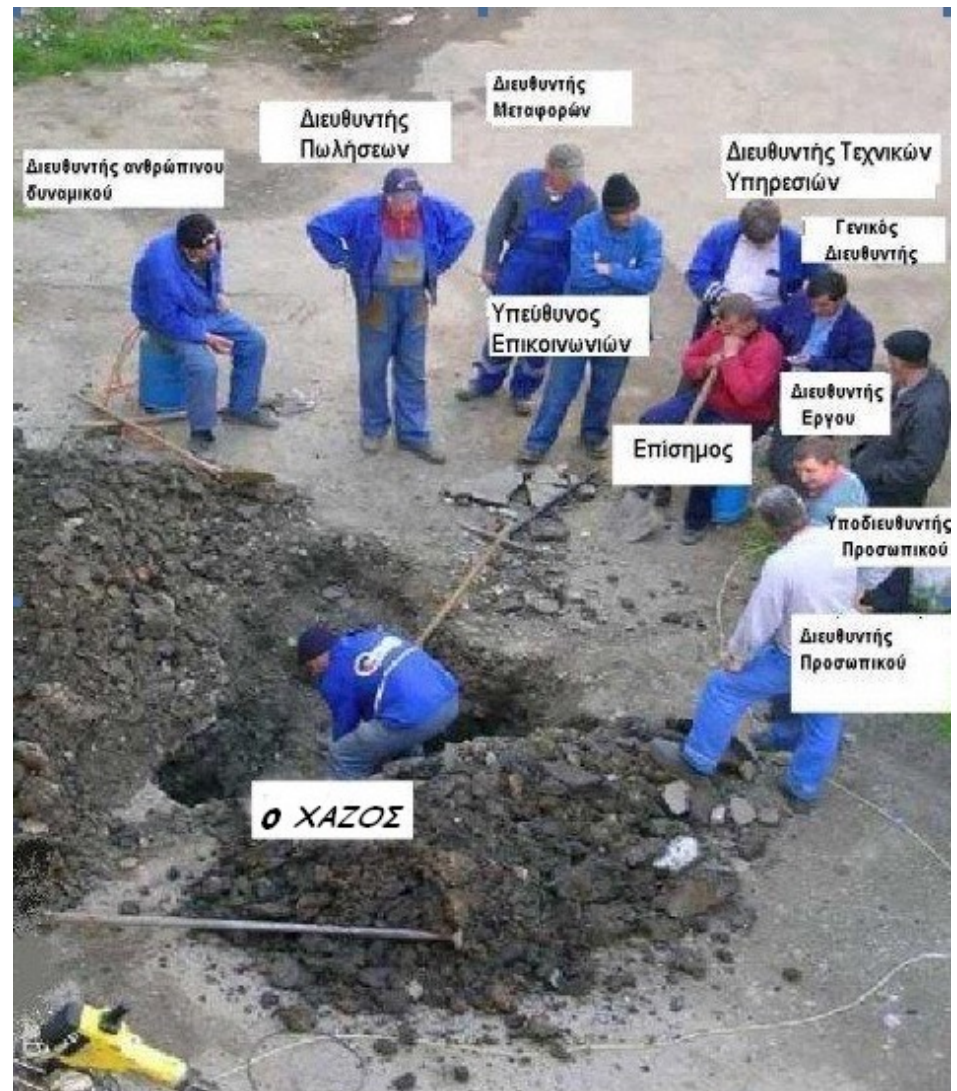


Ποιός εργάζεται πότε; Πόσο αργό το Ρολοί; – Άσκηση

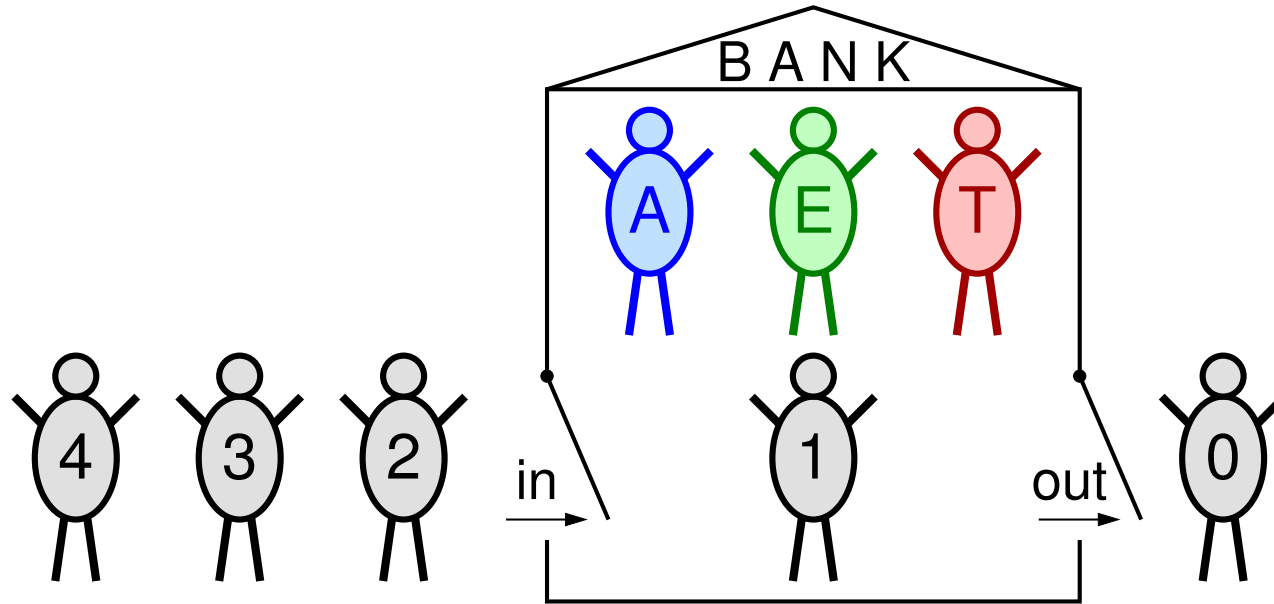


Ποιός εργάζεται πότε; Περί Σπατάλης Πόρων

- 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*)



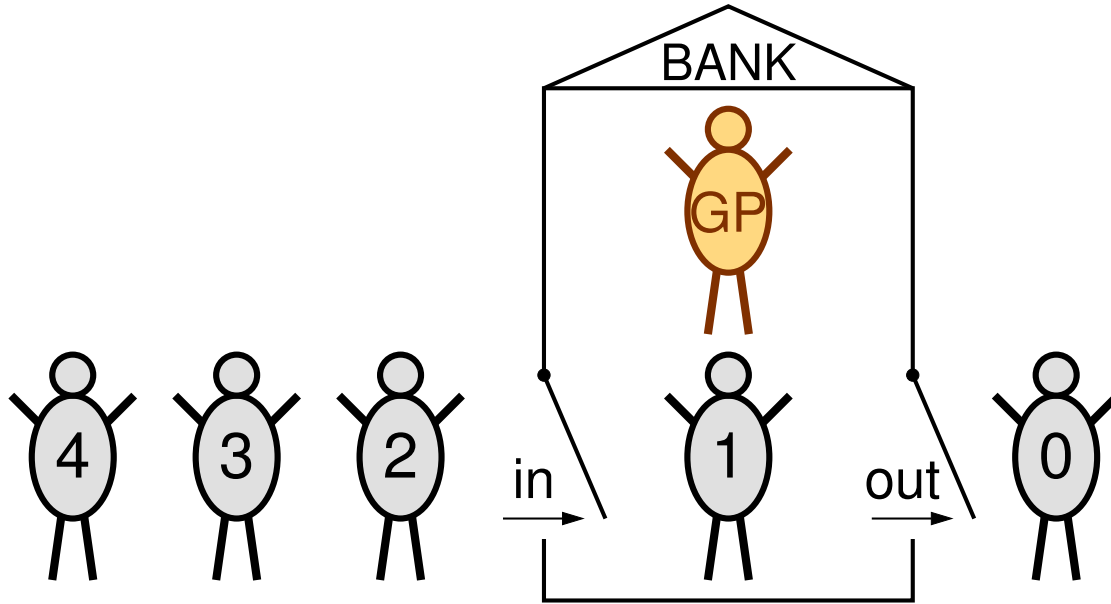
Γραμμές Παραγωγής με ειδικευμένους υποαπασχολ.



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

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

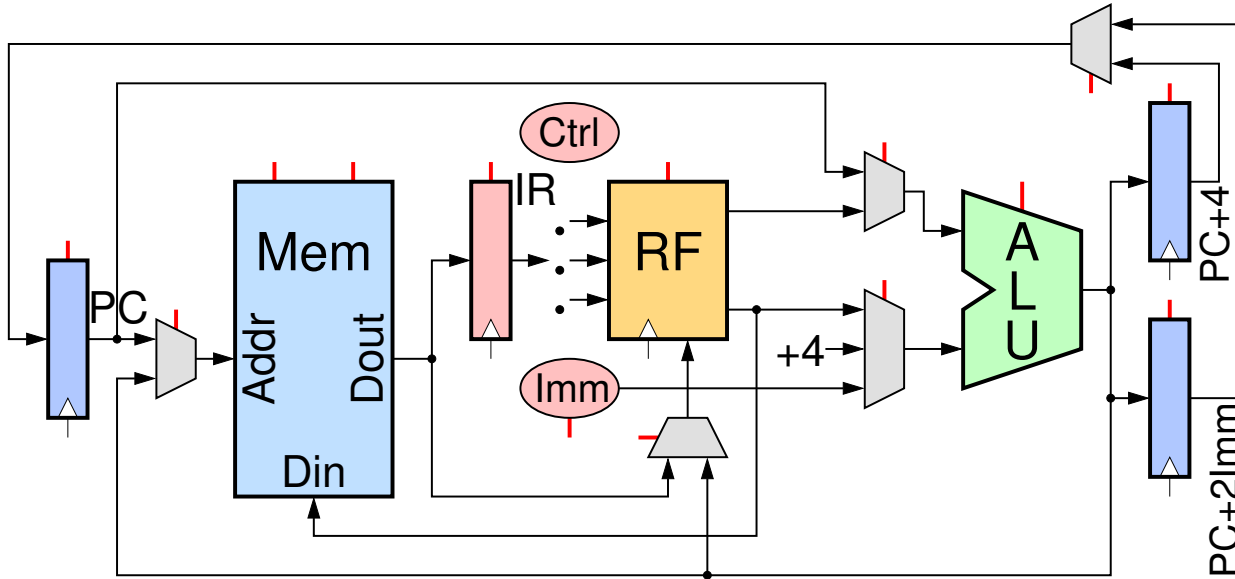
Εξοικονόμηση πόρων με Πόρους Γενικού Σκοπού



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

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

Παραλλαγή του υπολογιστή με εξοικονόμηση πόρων



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

