

Παραδείγματα ISA

(η βασική 32-bit αρχιτεκτονική RISC-V)

<https://mixstef.github.io/courses/comparch/>

Μ.Στεφανιδάκης



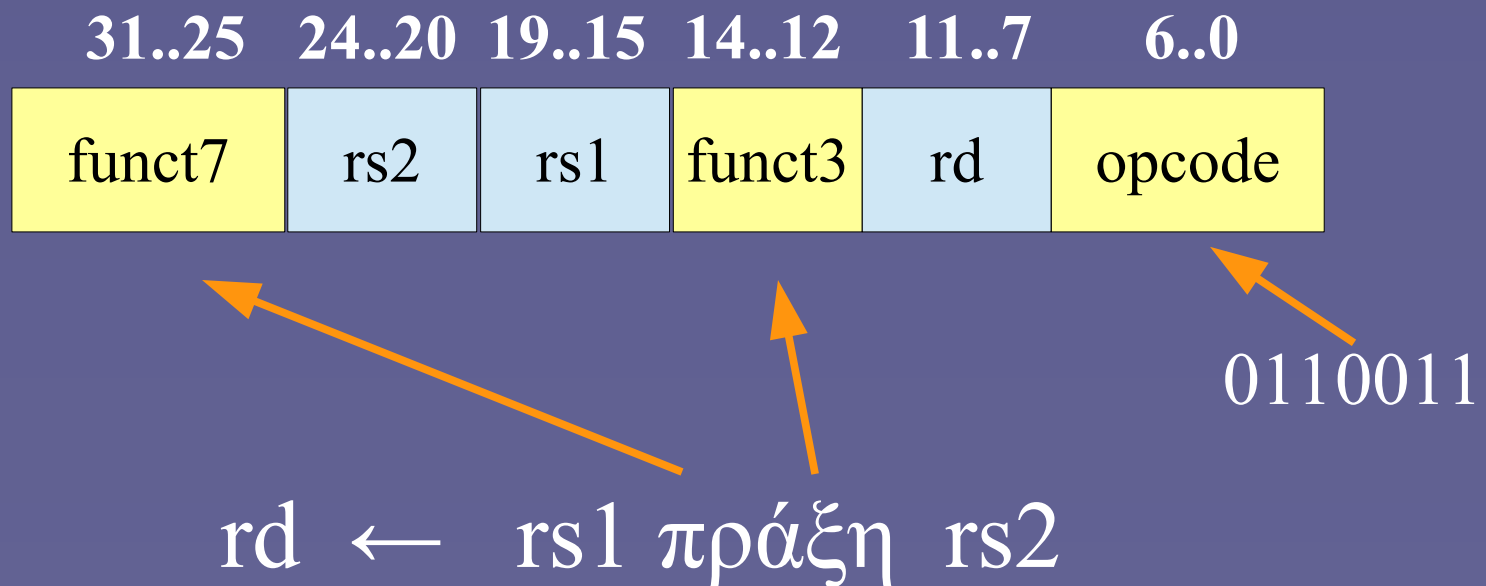
Η Αρχιτεκτονική Συνόλου Εντολών RISC-V

- Μια **ανοικτή** (open/free) αρχιτεκτονική συνόλου εντολών (ISA)
 - Ξεκίνησε από τον ακαδημαϊκό χώρο (Berkeley) αλλά τα τελευταία χρόνια γνωρίζει άνθιση στη βιομηχανία
 - Όλο και περισσότεροι κατασκευαστές σχεδιάζουν και υλοποιούν μονάδες επεξεργασίας με την αρχιτεκτονική συνόλου εντολών RISC-V
 - Δεν επιβάλλει συγκεκριμένη υλοποίηση αλλά έχει σχεδιαστεί για να διευκολύνει πιθανές υλοποιήσεις
- Η αρχιτεκτονική προδιαγράφει ένα **βασικό σύνολο** εντολών με πράξεις ακεραίων (32 ή 64 bits)
 - Και μία σειρά από **επεκτάσεις** (extensions)
 - Τη βασική **32-bit μορφή (RV32I)** θα δούμε στη συνέχεια

Κωδικοποίηση εντολών RV32I ISA

- 40 διαφορετικές εντολές
 - Κάθε εντολή έχει εύρος 32 bits
- Οι εντολές ομαδοποιούνται ως προς τη μορφή (instruction formats)
 - R, I, S, B, U και J Type
- 32 καταχωρητές (x0 έως x31) και επιπλέον ο program counter (pc)
 - Με εύρος 32 bits ο καθένας

Πράξεις μεταξύ καταχωρητών (R-type)

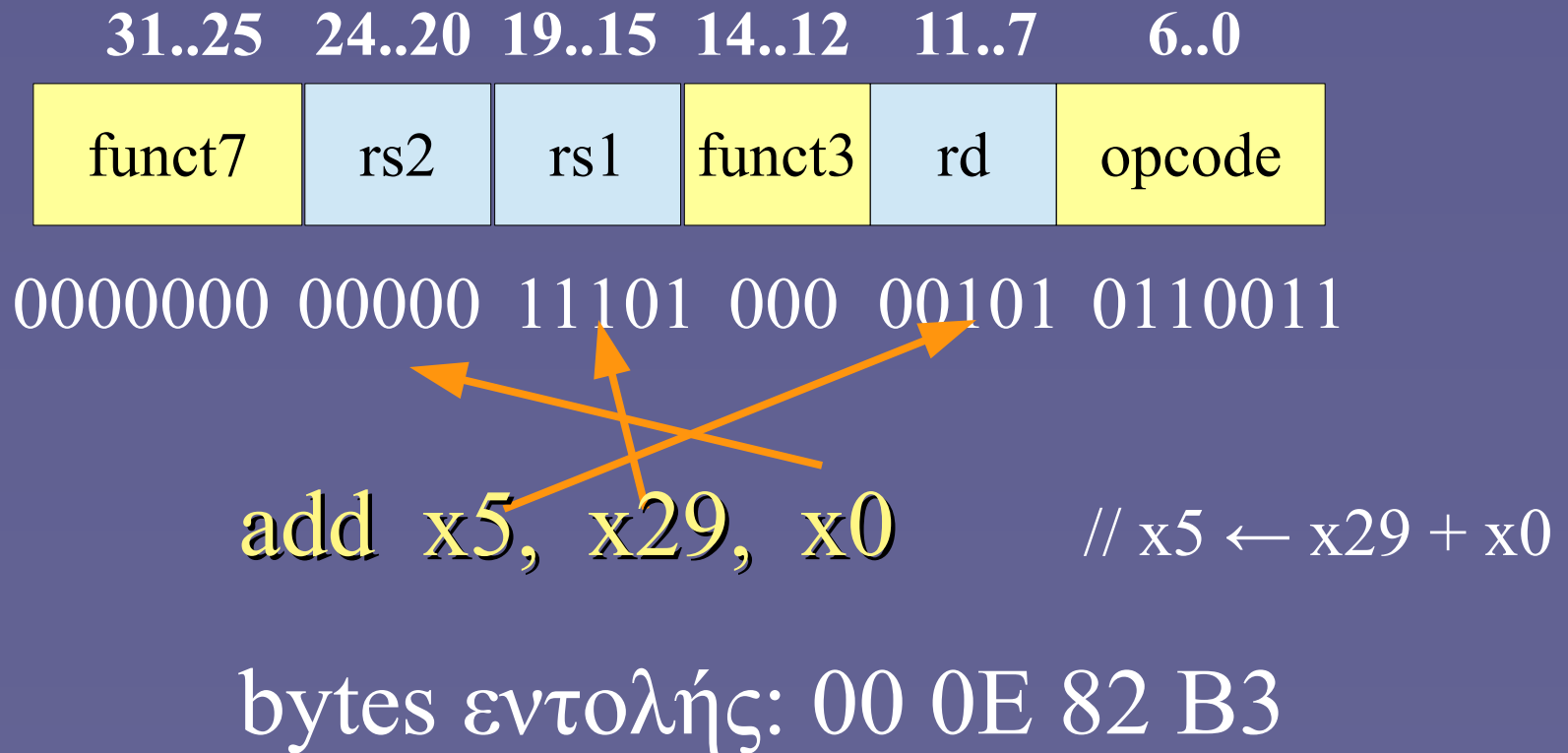


Επιλογή πράξης

funct7	funct3	πράξη
00000000	000	ADD
01000000	000	SUB
00000000	001	SLL
00000000	010	SLT
00000000	011	SLTU
00000000	100	XOR
00000000	101	SRL
01000000	101	SRA
00000000	110	OR
00000000	111	AND

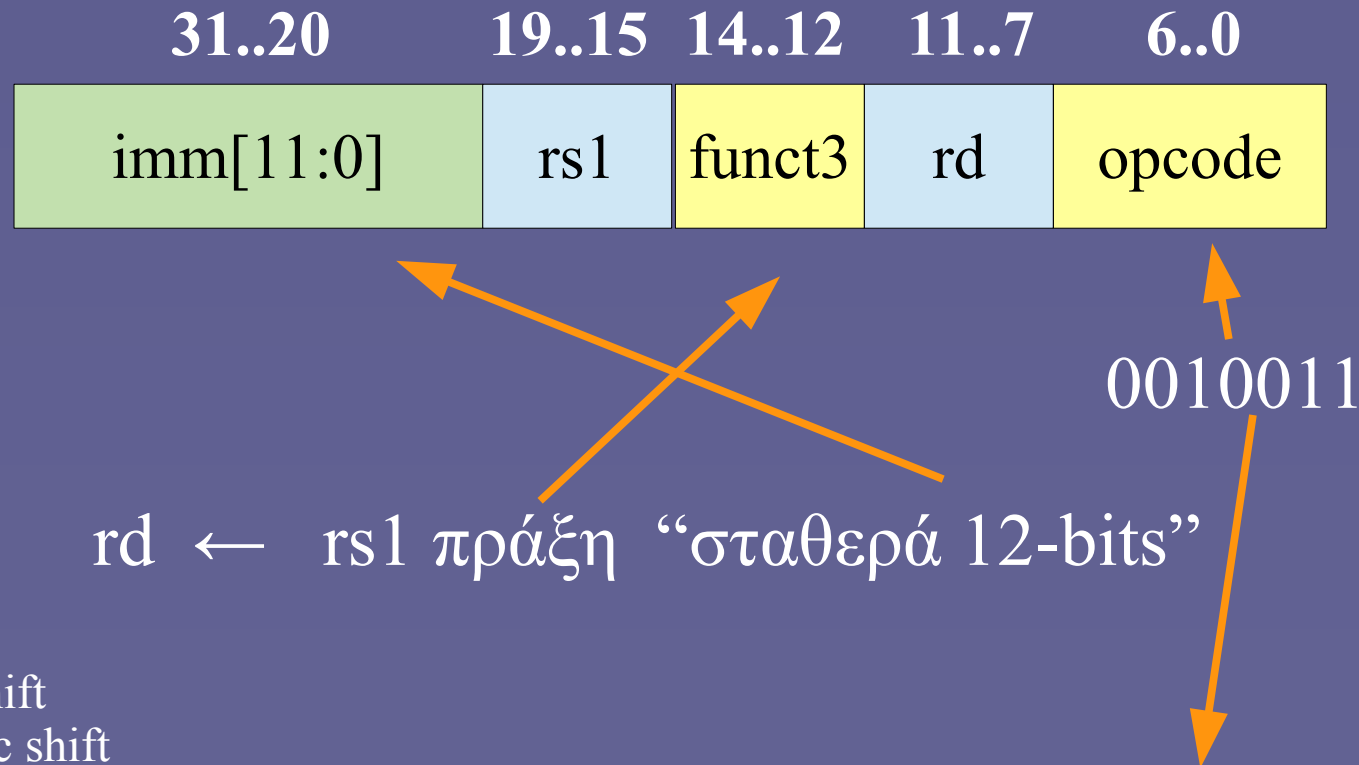
} $rd = (rs1 < rs2)?1:0$

Παράδειγμα

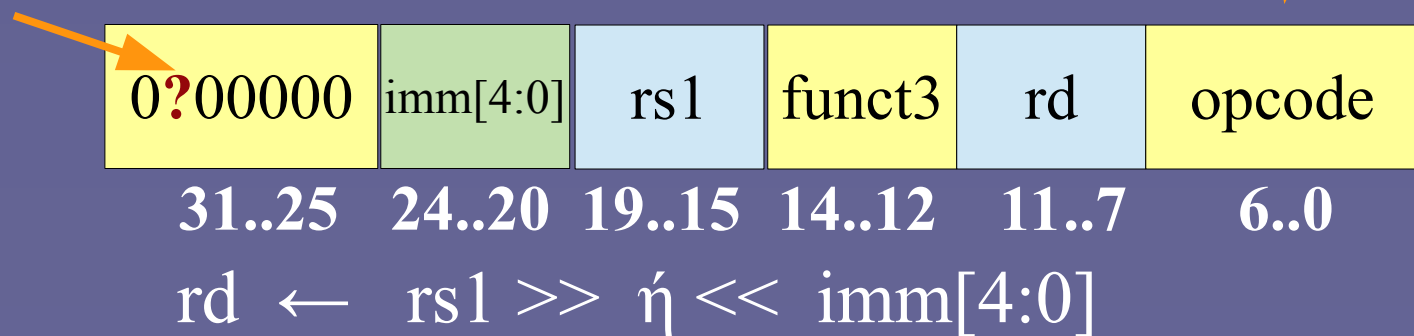


Πράξεις με σταθερή τιμή (I-type)

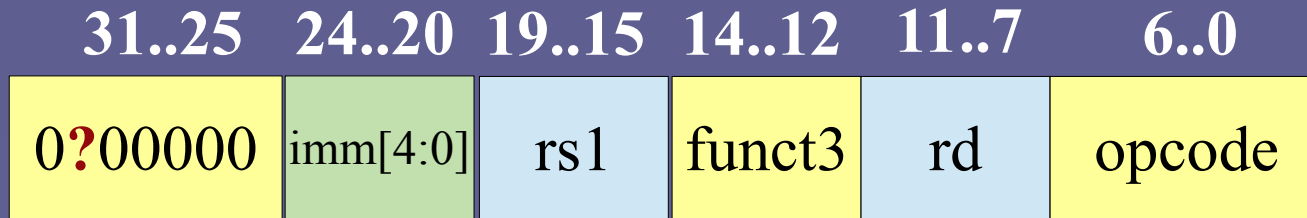
funct3	πράξη
000	ADDI
001	SLLI
010	SLTI
011	SLTIU
100	XORI
101	SRLI/SRAI
110	ORI
111	ANDI



0 = logical shift
1 = arithmetic shift



Παράδειγμα



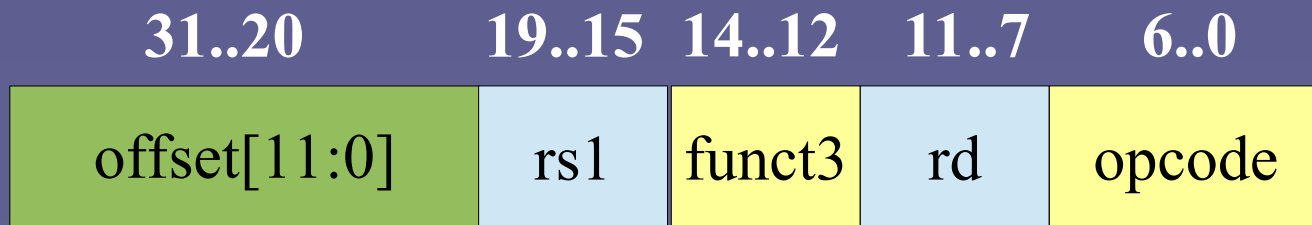
0100000 00110 00010 101 00100 0010011

srai x4, x2, 6

// $x4 \leftarrow x2 \gg 6$

bytes εντολής: 40 61 52 13

Load (I-type)



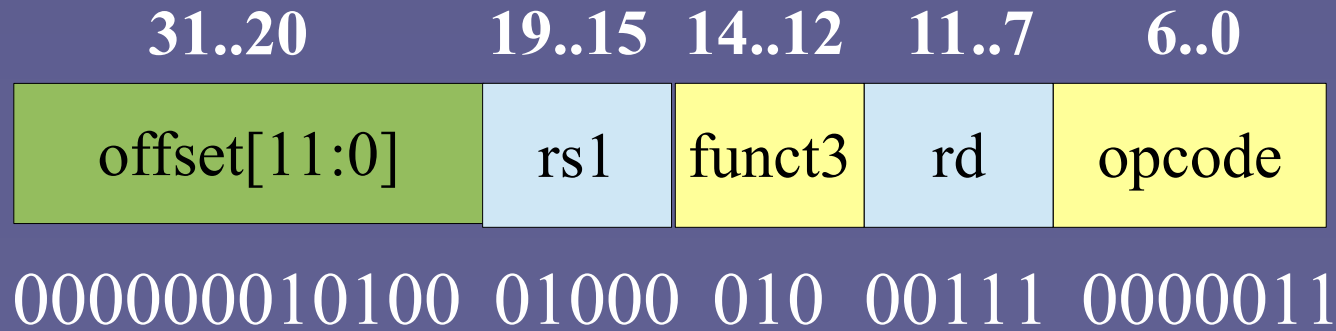
funct3	μεταφορά
000	LB (8 bits)
001	LH (16 bits)
010	LW (32 bits)
100	LBU (8 bits)
101	LHU (16 bits)

εύρος μεταφοράς

0000011

$rd \leftarrow \text{mem}[rs1 \pm \text{offset}]$

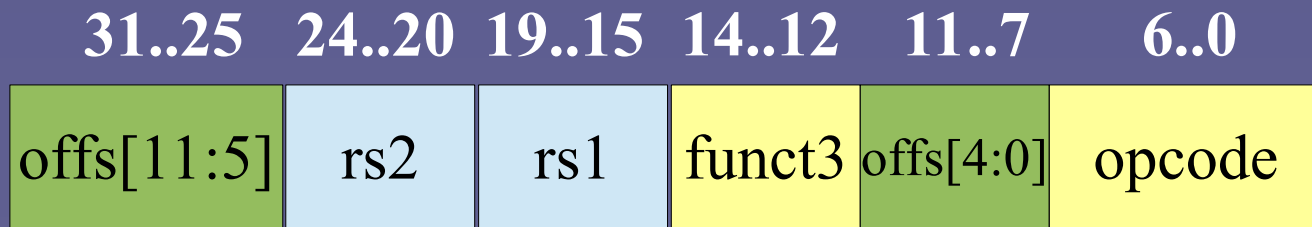
Παράδειγμα



`lw x7, 20(x8)` `// x7 ← mem.32[x8 + 20]`

bytes εντολής: 01 44 23 83

Store (S-type)



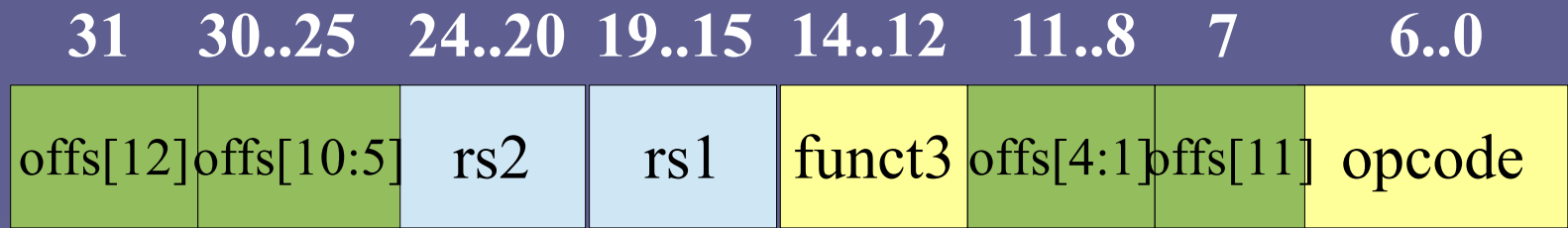
funct3	μεταφορά
000	SB (8 bits)
001	SH (16 bits)
010	SW (32 bits)

εύρος μεταφοράς

0100011

$rs2 \rightarrow mem[rs1 \pm offset]$

Conditional Branches (B-type)



funct3	σύγκριση (cmp)
000	BEQ (==)
001	BNE (!=)
100	BLT (<)
101	BGE (>=)
110	BLTU (<)
111	BGEU (>=)

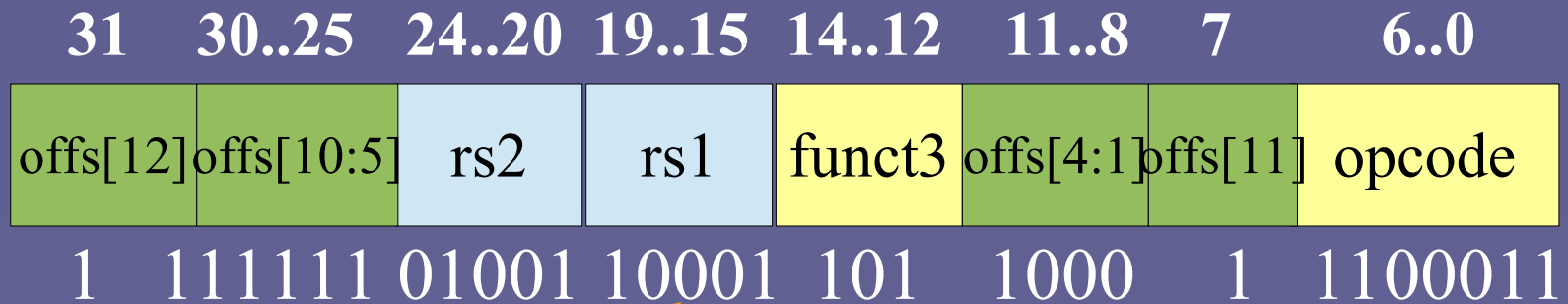
είδος σύγκρισης

1100011

$pc \leftarrow pc \pm \text{offset}$
if $rs1 \text{ } cmp \text{ } rs2$ is true

- Οι εντολές RISC-V είναι το λιγότερο 16 bits (2 bytes) → κάθε εντολή σε ζυγή διεύθυνση → το offs[0] είναι πάντα 0 και δεν αποθηκεύεται.
- Μετατόπιση $\pm 4\text{KiB}$ από τρέχουσα εντολή (branch)

Παράδειγμα



bge x17, x9, -16

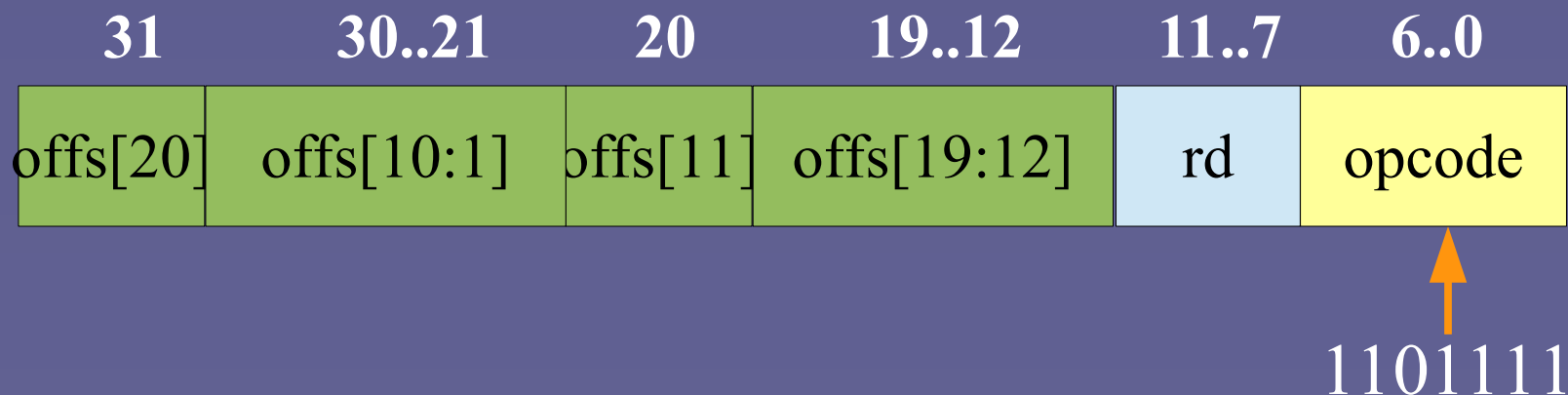
// if x17 \geq x9 then pc = pc-16

bytes εντολής: FE 98 D8 E3

Σημ.: -16 (σε 13 bits) = 1 1111 1111 000(0)

Δεν ξεχνάμε και το bit0 που θεωρείται πάντα 0

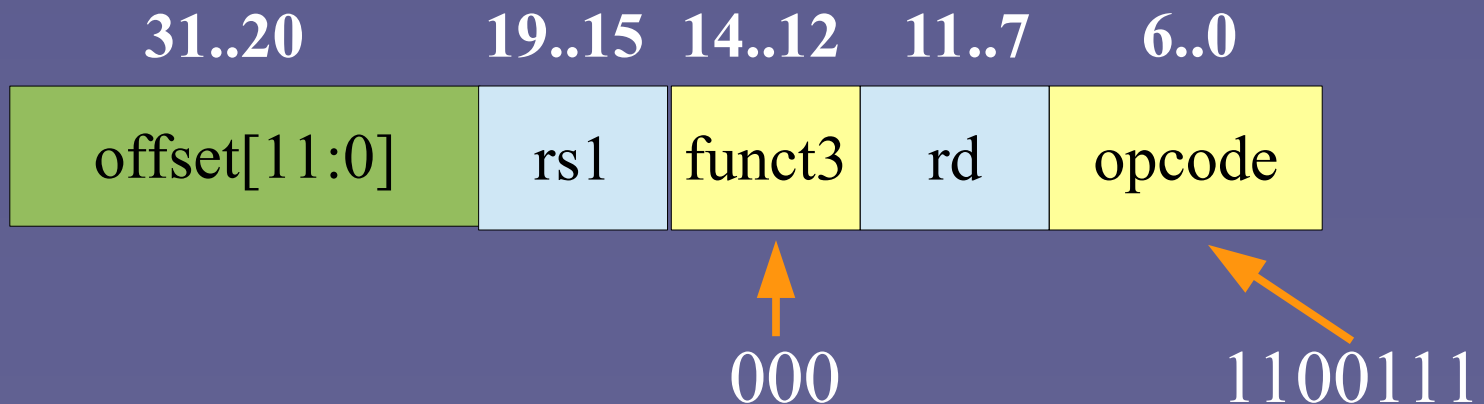
Jump and link (JAL, J-type)



$rd \leftarrow pc + 4$ (next instruction)
 $pc \leftarrow pc \pm \text{offset}$

Σημ.: offs[0] πάντα 0, δεν αποθηκεύεται → 21 bits για offset.
Μετατόπιση $\pm 1\text{MiB}$ από τρέχουσα εντολή (branch)

Jump and link register (JALR, I-type)



$rd \leftarrow pc + 4$ (next instruction)
 $pc \leftarrow rs1 \pm offset$

Σημ.: Εδώ προστίθεται το offset στον rs1 και στην τιμή που προκύπτει το bit0 γίνεται 0

Μεγάλες σταθερές (U-type)



0110111 (lui)
0010111 (auipc)

lui rd, imm // $rd \leftarrow imm \ll 12$

auipc rd, imm // $rd \leftarrow pc + (imm \ll 12)$

Σημ.: Εντολές για να βάλουμε σταθερές των 20 bits στο «πάνω» μέρος ενός καταχωρητή

Συνδυάζονται με άλλες εντολές που φορτώνουν τα 12 «χαμηλότερα» bits του καταχωρητή (π.χ. ADDI, αλλά προσοχή στην επέκταση προσήμου!)