Μεγέθη Εντολών στον RISC-V, η θέση του Opcode μέσα στην Εντολή, Πεδία μεταβλητού μεγέθους μέσα στην Εντολή

04a (§4.1-4.2) – 25 Φεβρουαρίου 2026 – Μανόλης Κατεβαίνης

Μέγεθος Εντολών RISC: συνήθως 32 bits

- Ρεπερτόρια CISC: Μεταβλητό μέγεθος εντολών
  - π.χ. VAX: 8, ή 16, ή 24, ή 32, ή 40, ή 48, ή… bits
- Πολλά Ρεπερτόρια RISC: Σταθερό μέγεθος εντ. = 32 bits
  - Σταθερό μέγεθος & θέση τελεστέων = απλότητα ⇒ ταχύτητα
  - Ευθυγράμμιση εντολών ⇒ πλήθος Bytes = δύναμη του 2
  - 16 bits είναι πολύ λίγα, 64 bits είναι περιττά πολλά ⇒ 32 bits…
- Βασικός RISC-V (*RV32I*): Όλες οι εντολές 32 bits
- Προαιρετικές επεκτάσεις: ακέραιο πολλαπλ. των 16 bits
- Προαιρ. επέκταση *C* (Compressed): εντολές 16 και 32 bits

RVC: Καινοτόμος τρόπος Code Compression

- Άλλα ρεπερτόρια: η κάθε εντολή έχει το δικό της, καθορισμένο μέγεθος, π.χ. ή 16 bits ή 32 bits (ή άλλο), καθεμία
- RISC-V: *όλες* οι εντολές υπάρχουν στα 32 bits (ή περισσοτ.)
- *Μερικές* από αυτές, υπάρχουν *και* σε *παραλλαγή* των 16 bits στην προαιρετική επέκταση C ("RVC")

$\Rightarrow$ Ο βασικός Compiler γεννά εκτελέσιμο κώδικα με μόνον 32-μπιτες εντολές του βασικού ρεπετορίου

- Εάν ένα τέτοιο εκτελέσιμο θέλουμε να το τρέξουμε σε επεξεργαστή RVC, τότε ο linker / loader αλλάζει σε 16-μπιτες όσες εντολές βρεί που έχουν και τέτοια παραλλαγή

Γιατί και πώς η συμπίεση κώδικα στον *RVC*

- Παρ' ότι λίγο δυσκολότερο hardware, ο RVC σημαντικός γιά:
  - embedded/IoT: πρέπει μικρά & φτηνά: μικρότερη μν. εντολών
  - ελάττωση της κατανάλωσης ενέργειας γιά ανάκληση εντολών
- Ποιές εντολές, κυρίως, χωρούν σε 16 bits:
  - μικρή σταθερή ποσότητα ή μικρό offset σε load/store
  - ένας από τους καταχ.: zero (x0) ή ret. addr. (x1) ή stack ptr. (x2)
  - ίδιος καταχωρητής προορισμού και πρώτος πηγής (π.χ. p=p+4)
  - καταχωρητές μεταξύ των 8 δημοφιλέστερων από τους 32.
- Συνήθως: 50-60% των αρχικών εντολών μπορούν να μικρύνουν ⇒ 25-30% ελάττωση μεγέθους προγράμματος

Θυμηθείτε / Σκεφτείτε:

- Μέγεθος Εντολών RISC συνήθως, βασικού RV
- Προαιρετική επέκταση *RVC* (Compressed):
  - Πόσες εντολές ⟶ τι μέγεθος;
  - Πώς το έκαναν οι άλλοι;
  - Πώς το κάνει ο *RVC*;
  - Γιατί έτσι;

  - Ποιοι είναι οι «συνήθεις ύποπτοι» για συμπίεση;

Θυμηθείτε / Σκεφτείτε:

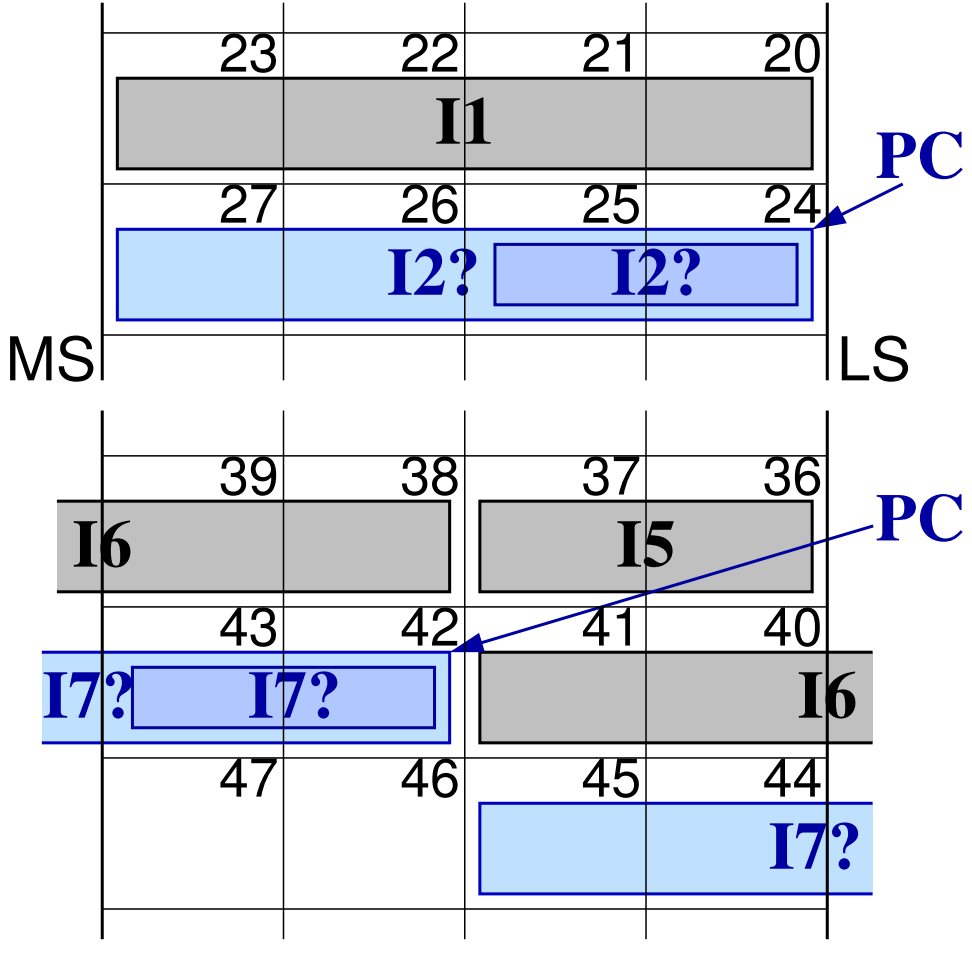
- Μέγεθος Εντολών RISC συνήθως, βασικού RV → 32 bits
- Προαιρετική επέκταση *RVC* (Compressed):
  - Πόσες εντολές → τι μέγεθος; → 50-60% → 16 bits
  - Πώς το έκαναν οι άλλοι; → κάθε εντολή ένα ορισμ. μεγ.
  - Πώς το κάνει ο *RVC*; → όλες 32 bits, μερικές **και** 16 bits
  - Γιατί έτσι; → βασικός Compiler όλες 32 bits,
    linker/loader προαιρετικά, όσες βρει → 16 bits
  - Ποιοι είναι οι «συνήθεις ύποπτοι» για συμπίεση;
    → μικρές σταθερές, x0, ra, sp, rd≡rs1, 8 popular reg's

Θέση του Opcode εντός της Εντολής

- Το μέγεθος της εντολής καθορίζεται εντός του opcode της
  - Πεδία τελεστέων: αυθαίρετες τιμές
- Μετά την **I1**, ποιό είναι το μεγεθος της εντολής **I2** @PC=24 ?
  - εάν η **I2** είναι 32 bits, ο opcode της δεν μπορεί να βρίσκεται στα Bytes 26 ή 27, διότι οι τελεστέοι της στα Bytes 24-25, που παίρνουν αυθαίρετες τιμές, θα μπορούσε να έμοιαζαν με κάποια άλλη, 16-μπιτη εντολή **I2** (διφορούμενο μεγ. εντ.)

⇒ *Opcode 32-μπιτων στα LS 16b*

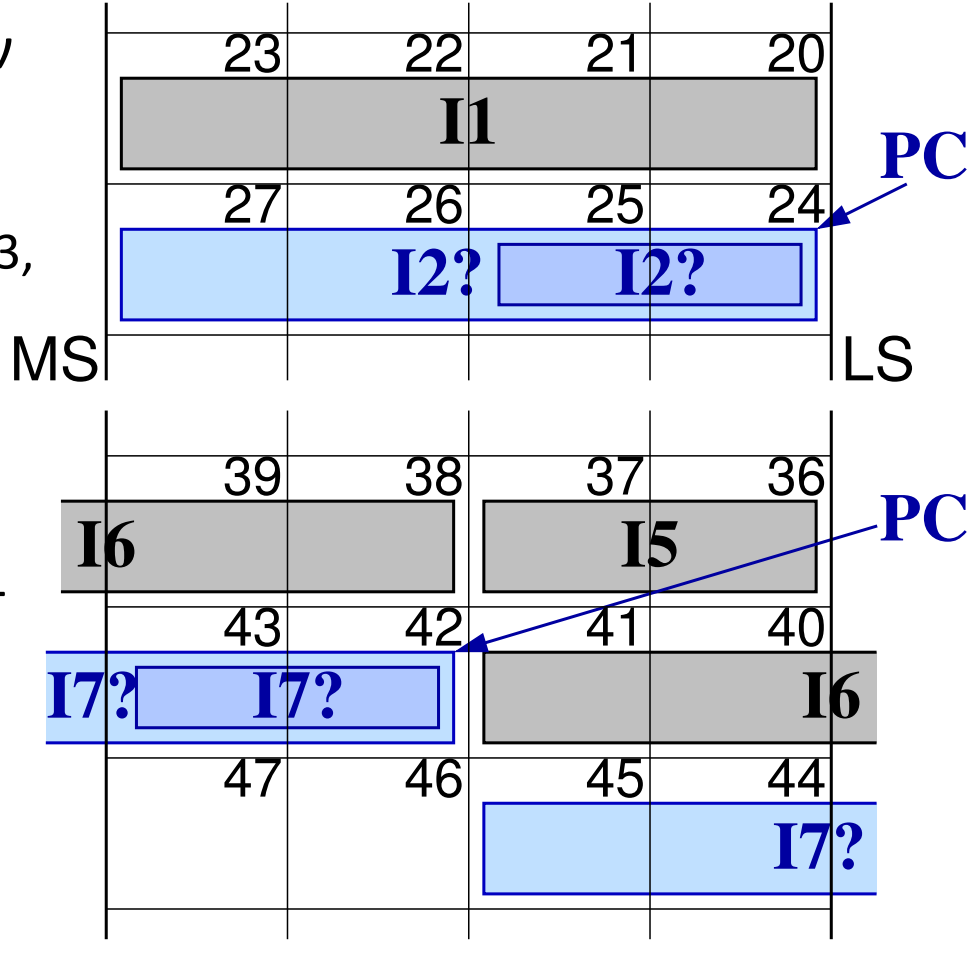
  - επειδή ο RISC-V είναι Little-Endian



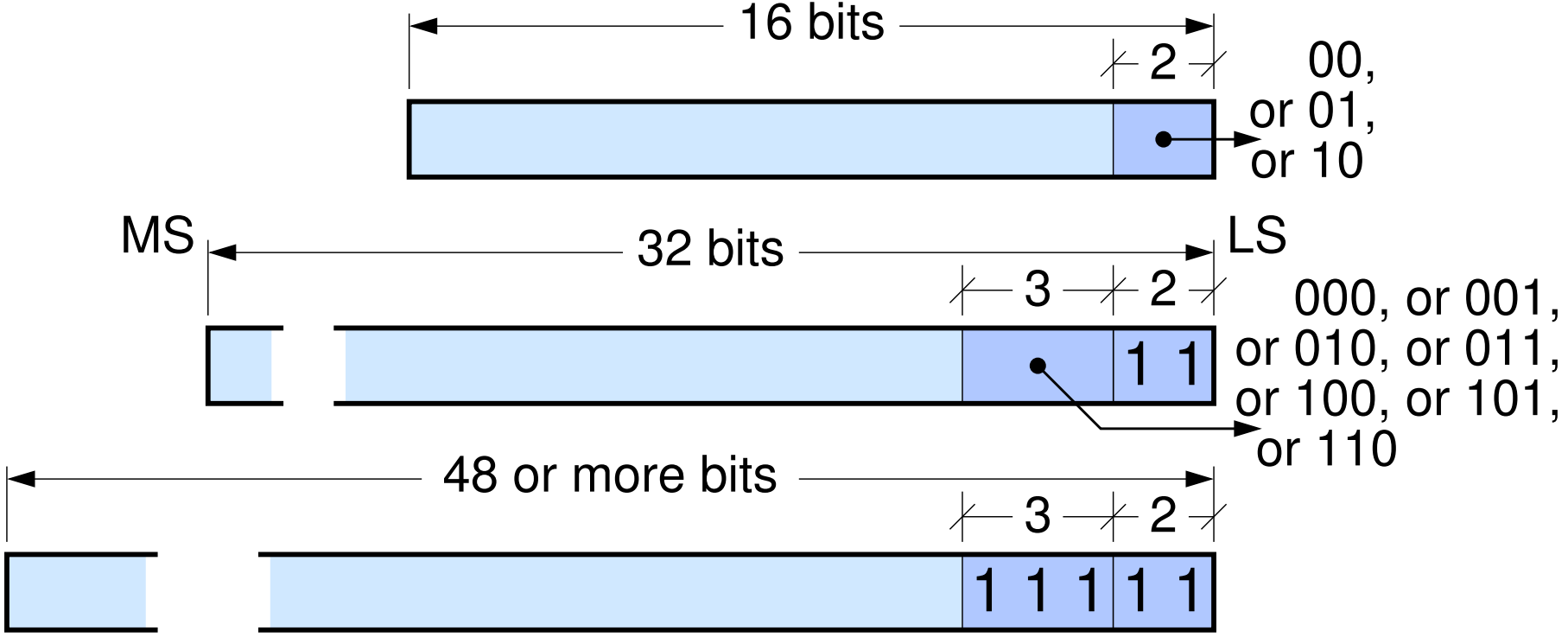
Ευθυγράμμιση Εντολών RV σε ακέρ. πολλαπλ. του **2**

- Λόγω της ύπαρξης και 16-μπιτων εντολών (π.χ. **I5**), οι 32-μπιτες μπορεί «σπασμένες» (π.χ. **I6**)
  - Εάν η **I7** είναι 32-μπιτη (Bytes 42, 43, 44, 45) το λιγότερο σημαντικό ήμισύ της είναι τα Bytes 42-43, άρα εκεί θα είναι το Opcode της

⇒ *Ευθυγράμμιση εντολών RISC-V σε 2-Byte boundaries* – **όχι** αναγκαστικά 4-Byte boundaries
  - Όμως σε επεξεργαστές χωρίς την προαιρετική επέκταση *C*, όλες οι εντολές είναι 32-μπιτες, άρα τις προτιμάμε σε 4-Byte boundaries



Μέγεθος εντολής: ορίζεται στα πρώτα 2 με 5 bits της



- Ο opcode στο λιγότερο σημ. (LS) άκρο της εντολής
- Τα πρώτα (LS) bits του opcode ορίζουν το μέγεθος της εντ.

Θυμηθείτε / Σκεφτείτε:

- Opcode αριστερά ή δεξιά στην εντολή;
- Γιατί (1 από 3);
- Γιατί (2 από 3);
- Γιατί (3 από 3);
- Ευθυγράμμιση Εντολών (*εγγυημένη*);
- Ευθυγράμμιση Εντολών (*ενδεχόμενη*);
- Ποια bits ορίζουν μέγεθος εντολής;
- Οι περισσότεροι συνδυασμοί για ποιες εντολές; γιατί;

Θυμηθείτε / Σκεφτείτε:

- Opcode αριστερά ή δεξιά στην εντολή; → δεξιά (LS)!
- Γιατί (1 από 3); → μπορεί να υπάρχουν και 16-μπιτες εντ.
- Γιατί (2 από 3); → ο Opcode δίνει το μέγεθος της εντολής
- Γιατί (3 από 3); → ο RISC-V είναι Little-Endian!
- Ευθυγράμμιση Εντολών (*εγγυημένη*); → ακέρ. πολ. του 2
- Ευθυγράμμιση Εντολών (*ενδεχόμενη*); → ακέρ. πολ. του 4
- Ποια bits ορίζουν μέγεθος εντολής; → τα πρώτα (δεξιά)
- Οι περισσότεροι συνδυασμοί για ποιες εντολές; γιατί;
  → για τις μικρότερες, δηλ. τις πιο «στριμωγμένες»

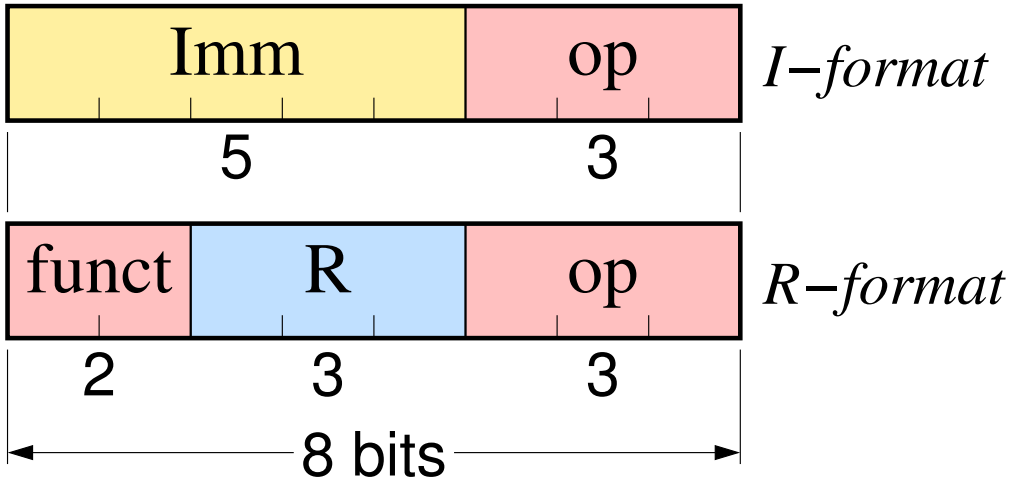
Πεδία μεταβλητού μεγέθους εντός 32-μπιτων εντολών

- Πόσο μεγάλες σταθερές (π.χ. addi) – πόσο μικρός Opcode?
  - π.χ. ο MIPS επιτρέπει 16-μπιτες σταθερές στις addi, lw, sw
  - συν 2 πεδία των 5 bits γιά τους 2 καταχωρ. = 26 bits μέχρι στιγμής
  - περισσεύουν 6 bits μόνον γιά τον opcode ⇒ 64 συνδυασμοί μόνο
  - εάν αυτό γίνονταν στον RISC-V, με τα LS bits του opcode όπως στην προηγούμενη διαφάνεια: $(7/8)\times 2^4 = 14$ συνδυασμοί μόνο…
- Πόσο μεγάλος Opcode γιά εντολές 3 καταχωρητών;
  - π.χ. η εντολή add έχει μόνον 3 καταχωρητές ως τελεστέους
  - 3 πεδία των 5 bits γιά τους 3 καταχωρ. = 15 bits μέχρι στιγμής
  - περισσεύουν 17 bits γιά τον opcode ⇒ 2^{17}=131.072 συνδυασμοί!

⇒ Opcode *μεταβλητού* μεγέθους!

Άσκ. 4.2: πόσοι Opcodes το ένα format, πόσοι το άλλο;

- Απλοϊκό παράδειγμα 8-μπιτων εντολών με έναν μόνο τελεστέο
- Opcode μεταβλητού μεγέθους: 3 ή 5 bits
- Opcode κομμένος σε δύο κομμάτια: "funct" (function) = επέκταση του "op"



- Πόσες εντολές με *I-format* και πόσες με *R-format*?
- Πόσοι **op** γιά το ένα και πόσοι γιά το άλλο;
  - μπορεί ο ίδιος **op** και γιά *I-format* και γιά *R-format*?
  - με οιαδήποτε, αυθαίρετη σταθερά **Imm**?