Τα Format Εντολών του βασικού RISC-V

04b (§4.3) – 3-5 Μαρτίου 2021 – Μανόλης Κατεβαίνης

Ιστορίες γιά τη θέση των πεδίων Καταχωρητών

- Στους RISC-I, RISC-II
  όπως στην Assembly:
  op, r1, r2, (*r3-funct* or *Imm*)
- Ανάγνωση Καταχωρητών:
  συνήθως τους *r2* και *r3*,
  αλλά ειδικά γιά την store τους *r2* και *r1*

- `add  r1, r2, r3`
- `addi r1, r2, Imm`
- `lw   r1, (r2)Imm`
- `sw   r1, (r2)Imm`

⇒ απαιτείται αποκωδικοποίηση του opcode και πολυ-πλέκτης πριν αρχίσει η ανάγνωση του "*r3* ή *r1*"

– αυτό καθυστερεί την έναρξη εκτέλεσης (ανάγν. 2 καταχ. πηγής)

– αναγνωρίστηκε εκ των υστέρων στο διδακτορικό του ομιλούντα

MIPS: Καταχωρητές **Πηγής** πάντα σε σταθερή θέση

- Στον MIPS:
  op, r1, r2, (rd-funct or *Imm)*
- Αμέσως μετά το Instr. fetch διαβάζονται οι καταχωρητές *r1* και *r2*, *ανεξαρτήτως* **op**
  - όταν υπάρχουν δύο καταχ. πηγής, αυτοί είναι *πάντα* οι r1, r2
- Αποκωδικοποίηση opcode εν παραλλήλω με αναγν. κατ.
  - καταχωρητής προορισμού άλλοτε ο *rd*, άλλοτε ο *r2*, αλλά αυτόν τον χρειαζόμαστε αργότερα κατά την εκτέλεση της εντολής
  - (John Moussouris, από το διδακτορικό του ομιλούντα)

```
• add  rd, r1, r2
• addi r2, r1, Imm
• lw   r2, (r1)Imm
• sw   r2, (r1)Imm
```

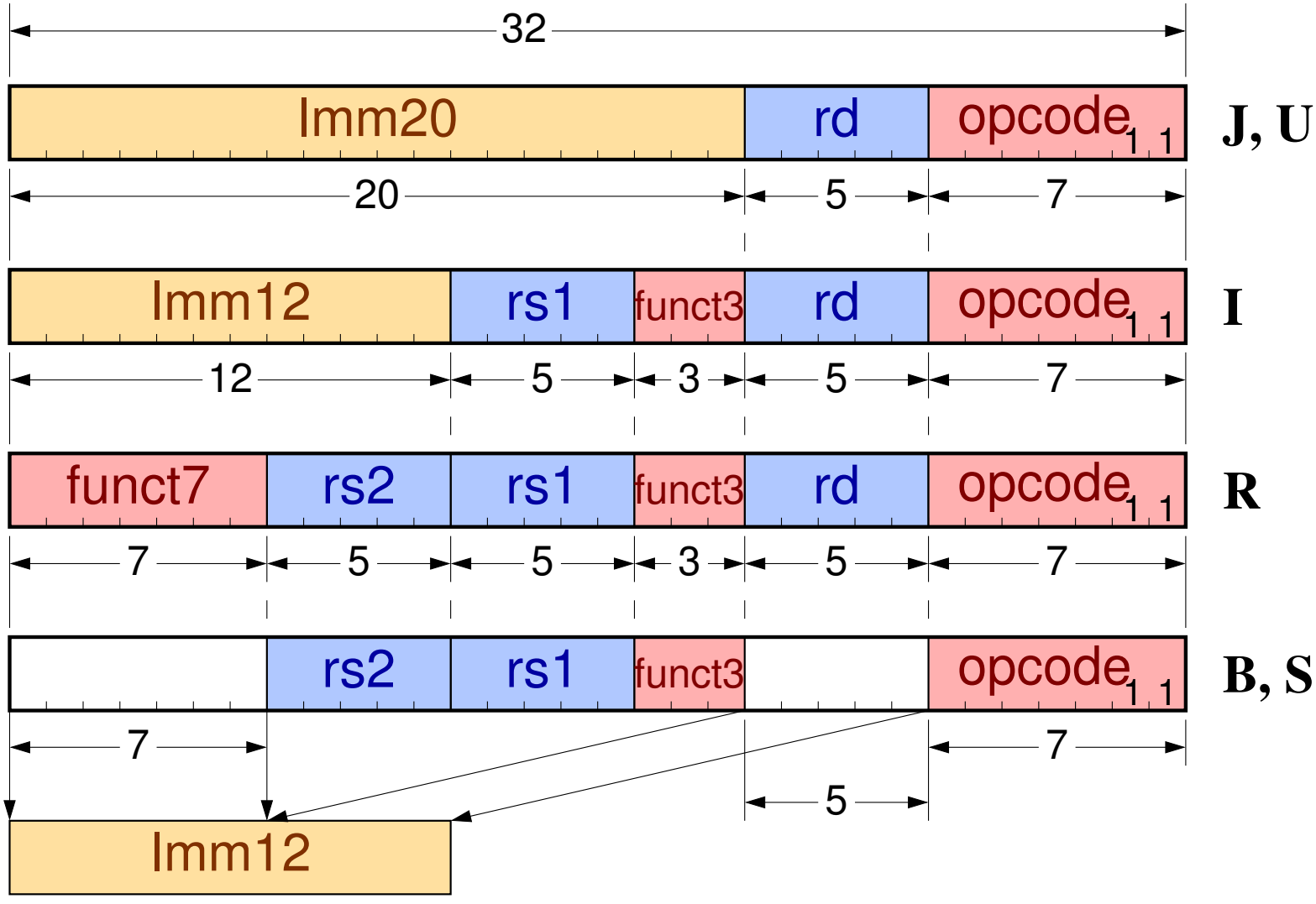
RISC-V: πάντα σε σταθερή θέση, όλοι οι καταχωρητές

- Οι σύγχρονοι επεξεργαστές: multiple-issue (superscalar): διαβάζουν 2 ή περισσότερες εντολές, εξετάζουν πόσες από αυτές ανεξάρτητες μεταξύ τους, και εκτελούν όσες είναι ανεξάρτητες εν παραλλήλω
- Έλεγχος εξαρτήσεων: είναι κάποιος καταχωρητής πηγής επόμενης εντολής ίδιος με καταχ. προορ. προηγούμενης;

⇒ Απαιτείται ταχεία σύγκριση πεδίων καταχωρητών

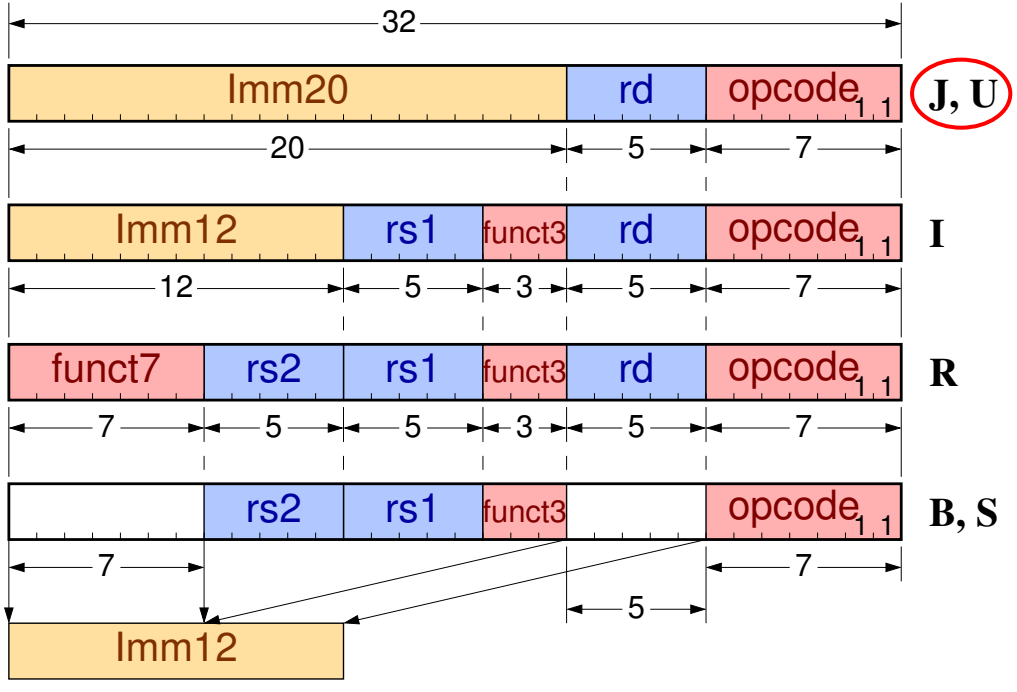
- Έχοντας όλα τα πεδία καταχωρητών πάντα σε σταθερές θέσεις, οι συγκρίσεις αρχίζουν *πριν* την αποκωδ. opcodes

Τα format Εντολών του RISC-V



J (Jump), **U** (Upper Immediate) format

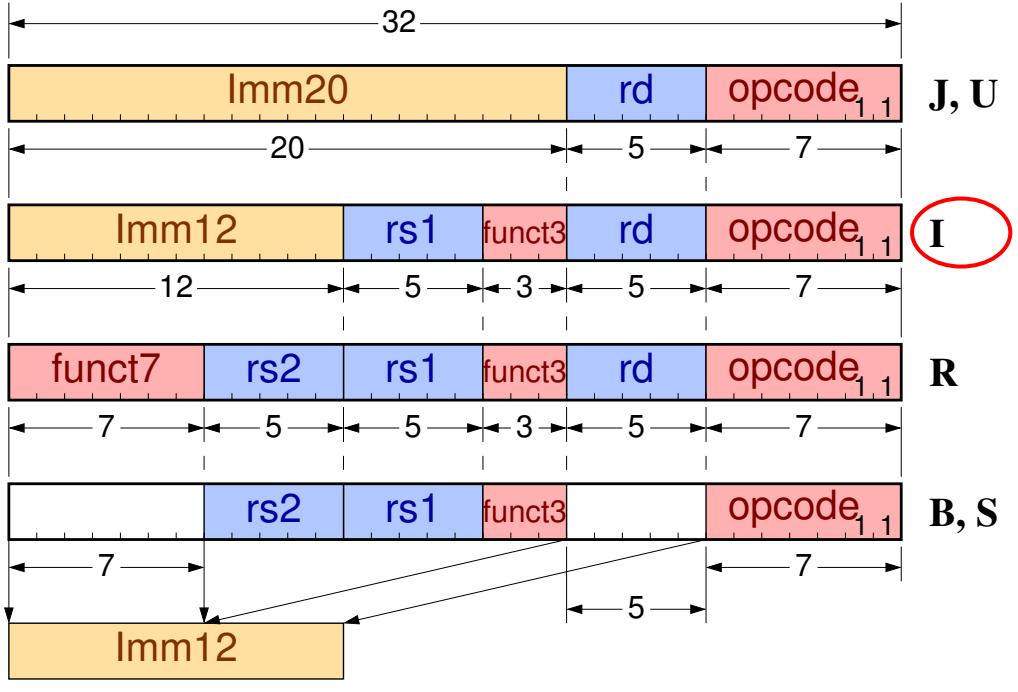
- Μόνο καταχωρητής προορισμού & (μεγάλη) σταθερά
- Κλήση διαδικασίας (`jal` – jump & link): μέγιστο δυνατό βελεκές
- Upper Immediate: *Imm20*, μαζί με *Imm12* επόμενης εντολής, κατασκευάζουν αυθαίρετη σταθερά 32 bits



- **J** format, **U** format: ίδια μεταξύ τους, εκτός ποιά bits είναι ποιά εντός του πεδίου *Imm20*
- Τρείς μόνο εντολές (*jal*, *lui*, *auipc*) – ξοδεύουν 3 από τους 28 διαθέσιμους συνδυασμούς του opcode γιά τις 32-μπιτες εντολές

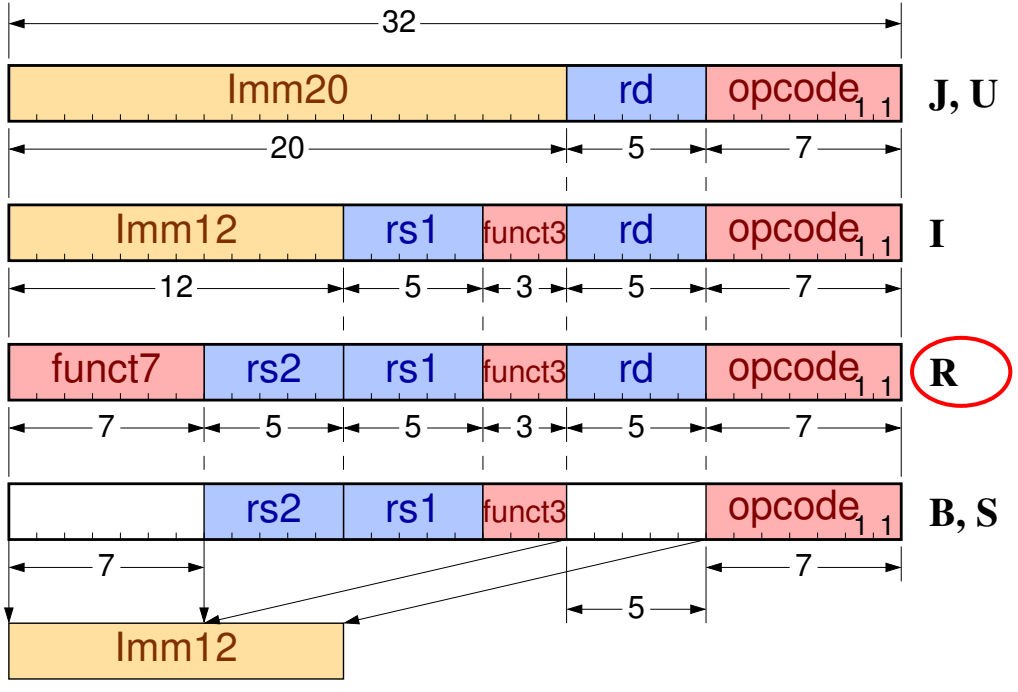
I (Immediate) format: Καταχωρητές και Σταθερά

- Παραδείγματα:
  ```
  addi rd, rs1, Imm12
  lw   rd, Imm12(rs1)
  ```
- Μόνον όταν ένας καταχωρ. πηγής, ο άλλος προορισμού
  - αλλιώς βλ. **B**, **S** format
- Πλάτος σταθεράς συμπληρωματικό με το *Imm20* του **U** format
- Πεδίο *funct3*: αποτελεί επέκταση του *opcode* σε συνολικά 10 bits
  - κομμένα μεταξύ τους, ώστε το *rd* να μείνει σε σταθερή θέση



R (Register) format: τρείς Τελεστέοι σε Καταχωρητές

- Παράδειγμα:
  `add rd, rs1, rs2`
- Οι καταχωρητές *rd*, *rs1* στις ίδιες θέσεις όπως και στα προηγούμενα format, **J**, **U**, **I**
- Πεδίο *funct7*: αποτελεί επέκταση των *opcode* και *funct3* σε συνολικά 17 bits
  - κομμένα μεταξύ τους, ώστε τα *rd* και *rs1* σε σταθερή θέση



B (Branch), **S** (Store) format: Καταχ. Πηγής & Σταθερά

- Παραδείγματα:
  `beq rs1, rs2, Imm12`
  `sw  rs2, Imm12(rs1)`
- Όμοιο με το **I** format, αλλά οι καταχωρητές είναι πηγής και οι δύο
- Η σταθερά *Imm12* κομμένη και αλλάζει θέση ώστε να μείνουν οι καταχωρητές *rs1*, *rs2* σε σταθερές θέσεις
- **B** format, **S** format: ίδια μεταξύ τους, εκτός ποιά bits είναι ποιά εντός του πεδίου *Imm12*