HY-225: Οργάνωση Υπολογιστών
Άνοιξη 2020

Τμ. Επ. Υπολογιστών
© Πανεπιστήμιο Κρήτης

Άσκηση 4:
Γλώσσα Μηχανής, Format Εντολών

κάντε την έως Τετάρτη 26 Φεβρουαρίου 2020 (βδ. 4.2) (από βδ. 2.2)

Βιβλίο: Το υλικό αυτό συγγενεύει στενά με τις σελίδες 81-89 του Αγγλικού βιβλίου (§2.5). Γιά τον MIPS, αυτό ήταν στις σελ. 134-142 του Ελληνικού βιβλίου, αλλά οι διαφορές με τον RISC-V είναι σημαντικές, άρα εδώ το Ελληνικό βιβλίο δεν βοηθά ιδιαίτερα.

4.1 Μέγεθος Εντολών, και η θέση του Opcode στον RISC-V:

Εσωτερικά, ο υπολογιστής λειτουργεί μόνο με δυαδικές τιμές (bits). Έτσι, γιά να μπορέσει να εκτελεστεί ένα πρόγραμμα Assembly πρέπει αυτό να μεταφραστεί σε *Γλώσσα Μηχανής*, δηλ. σε δυαδικά σύμβολα. Τη μετάφραση αυτή κάνει ένα πρόγραμμα, ο *Assembler* (γιά προγράμματα γραμμένα σε πολλαπλά αρχεία ή που καλούν βιβλιοθήκες, τη μετάφραση του κάθε αρχείου κάνει ο Assembler και τη συνένωση των επιμέρους μεταφρασμένων αρχείων την κάνει ο *Linker*). Όταν οι εντολές καταλήγουν στη Γλώσσα Μηχανής, πρέπει να καθοριστεί το μέγεθος και η θέση του κάθε "πεδίου" (field) μέα στην εντολή, δηλαδή του "opcode" που λέει **τι** θέλει να κάνει η εντολή, και των ορισμάτων (arguments) / τελεστέων (operands) που λένε πάνω **σε ποιούς** θα το κάνει αυτό. Αυτά πρέπει να οριστούν έτσι ώστε αφ' ενός να ταιριάζουν στο τι συνήθως κάνουν τα προγράμματα, και αφ' ετέρου να επιτρέπουν απλή και ταχεία ερμηνεία και εκτέλεση των εντολών από το hardware.

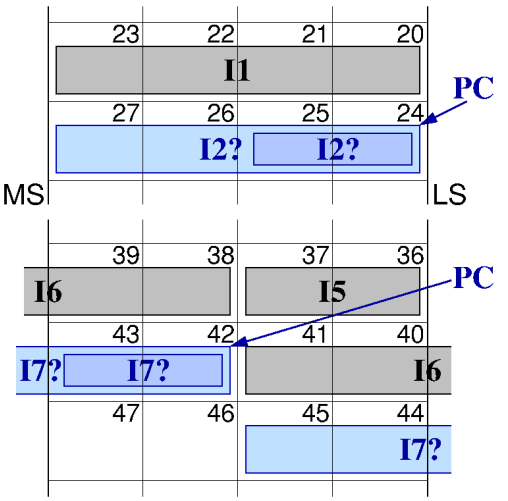
Στην οικογένεια RISC, το συντρηπτικά επικρατέστερο μέγεθος εντολής είναι **32 bits**, και πολλά ρεπερτόρια τύπου RISC έχουν *όλες* τις εντολές τους με μέγεθος 32 bits η καθεμία (έτσι είναι και ο MIPS), γιά λόγους ομοιομορφίας, άρα απλότητας, άρα ταχύτητας. Στις περιπτώσεις τέτοιων υπολογιστών, όλες οι εντολές είναι ευθυγραμμισμένες σε λέξεις των 32 bits, δηλαδή οι διευθύνσεις τους είναι ακέραια πολλαπλάσια του 4 (άρα τα 2 LS bits τους είναι πάντα 0). Και στον **βασικό** RISC-V (δηλαδή στο τμήμα του που είναι υποχρεωτικό γιά όλες τις υλοποιήσεις του RISC-V), όλες οι εντολές (αυτού του βασικού τμήματος) είναι επίσης μεγέθους **32 bits**.

Όμως, ο RISC-V περιλαμβάνει, μεταξύ των αρκετών *προαιρετικών* επεκτάσεων τις οποίες προβλέπει, και μία **προαιρετική επέκταση** με εντολές των 16 bits (την επέκταση "C" - Compressed Instructions), καθώς επίσης και προβλέψεις γιά ενδεχόμενες μελλοντικές (ή μη-προτυποποιημένες / ιδιωτικές - non-standard / proprietary) επεκτάσεις και με εντολές των 48 ή/και 64 ή/και 80... bits καθεμία (πάντοτε ακέραιο πολλαπλάσιο των 16 bits = 2 Bytes). Η κωδικοποίηση και αποκωδικοποίηση τέτοιων εντολών μεταβλητού μεγέθους, έχει πολλά κοινά με τους κώδικες μεταβλητού πλάτους (μήκους) που είδαμε στην §10.5 της Ψηφιακής Σχεδίασης, με τις εξής δύο βασικότερες διαφορές: (α) εδώ το "κβάντο" μεγέθους είναι τα 16 bits αντί του 1 bit τότε, και (β) εδώ η "όδευση" ανάγνωσης (κατ' αύξουσες διευθύνσεις) είναι Little Endian, άρα προχωρά από δεξιά προς τα αριστερά, σε αντίθεση με τότε που γράφαμε τα εισερχόμενα bits από αριστερά προς τα δεξιά.

Ειδικά γιά την επέκταση C των 16-μπιτων εντολών στον RISC-V αξίζει να σημειώσουμε τα εξής. Ο λόγος που εισήχθησαν αυτές είναι γιά όσους επιθυμούν να εξοικονομήσουν χώρο μνήμης γιά τα προγράμματα (και κρυφών μνημών εντολών): οι σχεδιαστές του RISC-V μέτρησαν ότι συχνά γύρω στο 50 με 60 % των εντολών μέσα σε ένα πρόγραμμα RISC-V μπορούν να μετατραπούν από 32 σε 16 bits καθεμία, προσφέροντας έτσι μιάν οικονομία γύρω στο 25 με 30 % στο συνολικό μέγεθος του προγράμματος στη μνήμη. Μιά πολύ ενδιαφέρουσα καινοτομία του RISC-V σε αυτό το θέμα είναι η εξής. Σε όλα τα άλλα ρεπερτόρια εντολών με μεταβλητό μέγεθος εντολών στο παρελθόν, η κάθε μία εντολή είχε ένα και συγκεκριμένο μέγεθος –άλλες "κοντές" και άλλες "μακριές". Αντιθέτως, στον RISC-V, *κάθε* 16-μπιτη εντολή υπάρχει και στα 32 bits, δηλαδή είναι ακριβώς ισοδύναμη με κάποια υπάρχουσα 32-μπιτη εντολή. Αυτό προσφέρει το εξής πρακτικό πλεονέκτημα: Οι compilers αρκεί να γεννάνε πάντα

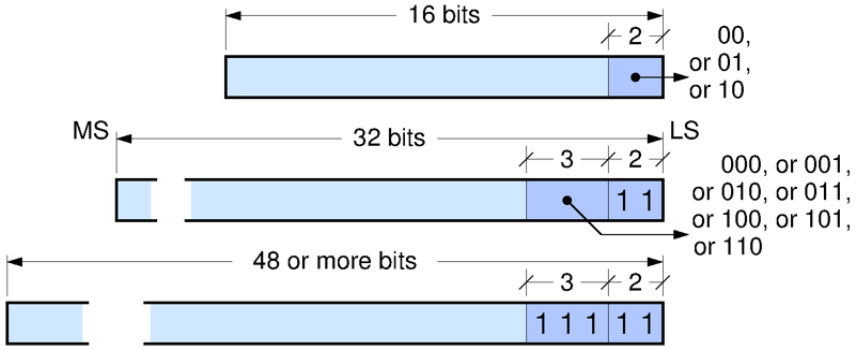
και μόνον 32-μπιτες εντολές RISC-V. Εάν και όταν ένα πρόγραμμα που έχει γίνει compiled έτσι θελήσουμε να τρέξει σε κάποιον επεξεργαστή ο οποίος περιλαμβάνει την προαιρετική επέκταση C, τότε ο Assembler και ο Linker μπορούν να αλλάξουν κάμποσες από τις 32-μπιτες εντολές σε 16-μπιτες –όσες από αυτές τυχαίνει να έχουν 16-μπιτο ισοδύναμο, μειώνοντας έτσι το μέγεθος του προγράμματος.

Δεδομένου ότι όλα τα προβλεπόμενα και επιτρεπτά μεγέθη εντολών του RISC-V είναι ακέραια πολλαπλάσια των 16 bits (2 Bytes), προκύπτει ότι, εάν ένα πρόγραμμα ξεκινά από διεύθυνση μνήμης που είναι ακέραιο πολλαπλάσιο του 2, τότε όλες οι εντολές του θα βρίσκονται σε διευθύνσεις που θα είναι επίσης ακέραια πολλαπλάσια του 2, άρα όλες τους ευθυγραμμισμένες σε φυσικά όρια half-words (2 Bytes). Έτσι, οι εντολές άλματος/διακλάδωσης του RISC-V που θα δούμε στην επόμενη σειρά ασκήσεων, θεωρούν, επιβάλουν, και αξιοποιούν το ότι ο PC (Program Counter - Μετρητής Προγράμματος) έχει πάντοτε τιμή ακέραιο πολλαπλάσιο του 2 (δηλαδή που λήγει σε 0, στο δυαδικό). Από την άλλη, ένας επεξεργαστής RISC-V που δεν περιλαμβάνει την προαιρετική επέκταση C, και άρα δεν έχει 16-μπιτες (ούτε 48-μπιτες) εντολές, θα έχει σαφές όφελος σε ταχύτητα και απλότητα ευθυγραμμίζοντας όλες τις εντολές του σε ακέραια πολλαπλάσια του 4, δηλαδή σε φυσικά όρια words (4 Bytes).



Η απαίτηση γιά δυνατότητα παρουσίας και 16-μπιτων εντολών στον RISC-V επιβάλει περιορισμούς στο **πού** μέσα στην εντολή (σε γλώσσα μηχανής) πρέπει να τοποθετηθεί ο opcode, δηλαδή ο δυαδικός κωδικός που ορίζει τι πρέπει να κάνει η εντολή, άρα και εάν αυτή είναι 16-μπιτη ή 32-μπιτη (ή 48, ή 64, κλπ) εντολή. Αυτό φαίνεται στο παράδειγμα στο σχήμα δεξιά, όπου το κάθε "κουτάκι είναι 1 Byte της (32-μπιτης) μνήμης, και η διεύθυνση του κάθε Byte αναγράφεται μέσα του και πάνω δεξιά· τα μακρουλά και σκιασμένα παραλληλόγραμμα είναι εντολές. Στο πρώτο μέρος του σχήματος, υποτίθεται ότι ο επεξεργαστής μόλις τελείωσε την εκτέλεση της (γκρίζας) εντολής I1, από τη διεύθυνση 20, η οποία τυχαίνει να ήταν 32-μπιτη (4 Bytes), και άρα ο PC τώρα είναι 24 και ο επεξεργαστής πρέπει να διαβάσει την εντολή στη διεύθυνση 24, να καταλάβει τι είναι και τι λέει αυτή, και στη συνέχεια να την εκτελέσει. Η εντολή αυτή στη διεύθυνση 24, έστω η I2, μπορεί να είναι είτε 16-μπιτη είτε 32-μπιτη. Εάν είναι 16-μπιτη, τότε αυτή αποτελείται από τα Bytes 24 και 25. Εάν είναι 32-μπιτη, τότε αποτελείται από τα Bytes 24, 25, 26, και 27. Άρα ο επεξεργαστής πρέπει να μπορεί να μάθει το μέγεθος της I2 κοιτώντας αποκλειστικά και μόνον τα Bytes 24 και 25 –όχι τα 26 ή 27, αφού τα 26 και 27 μπορεί να είναι μέρος άλλης εντολής, της I3, στην περίπτωση που η I2 είναι 16-μπιτη. Δεδομένου ότι ο RISC-V είναι Little Endian (§3.3), προκύπτει ότι τα Bytes 24 και 25 είναι τα δύο **δεξιά** (λιγότερο σημαντικά - LS) μέσα στην 32-μπιτη λέξη 24-25-26-27, άρα μέσα στις 32-μπιτες εντολές (σε περίπτωση που η I2 είναι 32-μπιτη), **πρέπει** ο opcode να βρίσκεται στο **δεξί ήμισυ** της εντολής (σε αντίθεση π.χ. με το τι ισχύει στον MIPS, ή στον απλό υπολογιστή της Ψηφιακής μας Σχεδίασης, όπου ο opcode ήταν στο "αριστερό" (MS) άκρο της εντολής).

Τα ίδια ισχύουν και στην λίγο πιό περίπλοκη περίπτωση που δείχνει το δεύτερο μέρος του σχήματος, εκεί που η επόμενη προς εκτέλεση εντολή είναι η I7 από τη διεύθυνση 42, όπου το 42 είναι μεν ακέραιο πολλαπλάσιο του 2, όχι όμως και του 4. Εδώ, ο επεξεργαστής έτυχε να έχει μόλις εκτελέσει τη 16-μπιτη εντολή I5, και αμέσως μετά την 32-μπιτη I6 που (αναγκαστικά) είναι "σπασμένη" σε δύο κομάτια, μέσα σε δύο διαφορετικές λέξεις μνήμης το καθένα. Η εντολή I7 δεν ξέρουμε ακόμα εάν είναι 16-μπιτη (Bytes @ 42 και 43) ή εάν είναι 32-μπιτη (Bytes @ 42, 43, 44, και 45). Άρα η πληροφορία γιά το μέγεθός της –που παραδοσιακά βρίσκεται μέσα στον opcode της– πρέπει να βρίσκεται μέσα στα Bytes 42 και 43 της εντολής, ακόμα και στην περίπτωση που αυτή είναι 32-μπιτη, άρα στα δύο λιγότερο σημαντικά ("δεξιά") Bytes της (έστω και εάν στο σχήμα το δεξιό κομάτι της 32-μπιτης I7 είναι "σπασμένο" και τοποθετημένο στην προηγούμενη λέξη μνήμης).



Δοθέντος του περιορισμού ο opcode να βρίσκεται "προς τα δεξιά" μέσα στην εντολή, ο RISC-V τοποθετεί τον βασικό του opcode στο *άκρο δεξιά* μέσα στην εντολή, δηλαδή στα λιγότερο σημαντικά bits της (μιλάμε γιά τον "βασικό opcode" επειδή αρκετές εντολές έχουν και δεύτερη ή και τρίτη επέκτασή του, υπό το όνομα "function code(s)"). Μέσα στον βασικό opcode, τα "πρώτα" (άρα δεξιότερα, LS) bits είναι αυτά που καθορίζουν το μέγεθος της εντολής, όπως φαίνεται στο σχήμα: Όταν τα πρώτα (δεξιότερα) δύο bits είναι είτε 00, είτε 01, είτε 10, τότε η εντολή είναι 16-μπιτη, ενώ εάν αυτά τα πρώτα δύο bits είναι 11, τότε η εντολή είναι των 32 ή περισσοτέρων bits. Στην τελευταία αυτή περίπτωση, με τα δύο πρώτα (δεξιότερα) bits ίσα με 11, τότε εάν τα τρία επόμενα (προς τα αριστερά) bits είναι 111 θα πρόκειται γιά "μεγάλη" εντολή, των 48 ή περισσοτέρων bits, ενώ εάν αυτά τα τρία "επόμενα" bits σχηματίζουν έναν οιονδήποτε από τους υπόλοιπους 7 συνδυασμούς (000, 001, ..., 110) τότε πρόκειται γιά μιά "κλασσική", 32-μπιτη εντολή.

Γιά περισσότερες λεπτομέρειες σε όλα αυτά τα θέματα, μπορείτε να κοιτάξετε τις σελίδες 7-10, 97-100, 129-130, και 141-147 του εγχειρίδιου (manual) του RISC-V, με τρέχουσα νεότερη έκδοση την 20191213: riscv-spec-20191213.pdf που ευρίσκεται διαθέσιμο γιά "κατέβασμα" όπως είπαμε και στην 1η σειρά ασκήσεων από τη σελίδα: riscv.org/specifications/

Άσκηση 4.2: Γλώσσα Μηχανής και Κώδικες Μεταβλητού Πλάτους

Κατ' αναλογία προς τις εντολές μεταβλητού μεγέθους (πλάτους), και μέσα στην κάθε εντολή χρησιμοποιούνται πεδία μεταβλητού πλάτους, και πρώτ' απ' όλα ο ίδιος ο opcode, ο οποίος μάλιστα μπορεί και να είναι "κομένος" σε δύο ή και τρία κομάτια, "σπαρμένα" σε διάφορες (μη συνεχόμενες) θέσεις μέσα στην εντολή. Σαν προετοιμασία γιά το θέμα αυτό στον (βασικό) RISC-V, κάντε την εξής απλοποιημένη άσκηση –απλοποιημένη κατά το ότι θεωρούμε ότι οι εντολές έχουν 8 μόνον bits, και περιέχουν ένα μόνον άλλο πεδίο πέραν του opcode (επεκτεταμένο μέσω του "function code") –αλλά αυτό το άλλο πεδίο είναι άλλοτε στενό και άλλοτε φαρδύτερο.

Τον βασικό opcode ας τον γράφουμε εδώ "`op`" και την επέκτασή του (function code) ας την γράφουμε "`funct`". Γιά να μην παιδευόμαστε με μεγάλο πλήθος εντολών, θα χρησιμοποιήσουμε ένα δικό μας, φανταστικό format εντολών, ενός φανταστικού υπολογιστή, του RV_8. Ολες οι εντολές του RV_8 έχουν μέγεθος 8 bits. Ο RV_8 έχει μόνο 8 καταχωρητές, τους x0, x1, ..., x7, και οι σταθερές του ποσότητες (immediates) μπορούν να είναι μόνο οι 32 μη-αρνητικοί (unsigned) ακέραιοι 0, 1, 2, ..., 31. Οι εντολές του RV_8 έχουν μόνον έναν τελεστέο – είτε καταχωρητή είτε σταθερή ποσότητα immediate– και έχουν μόνο δύο format, τα εξής:

I-format:

- **`op`** (3 LS bits): ο (μοναδικός) opcode,
- **`Imm`** (5 MS bits): η σταθερά - τελεστέος.

R-format:

- **`op`** (3 LS bits): πρώτο μέρος του opcode,
- **`R`** (3 επόμενα bits): ο καταχωρητής - τελεστέος,
- **`funct`** (2 MS bits): δεύτερο μέρος του opcode.

4(a): Εστω ότι "ξοδεύουμε" και τους οκτώ (8) διαθέσιμους συνδυασμούς του πεδίου op γιά 8 εντολές I-format, τις εντολές **`ki, li, mi, ni, pi, qi, ri, si`**. Σ' αυτή την περίπτωση, μπορούμε να έχουμε καμία εντολή R-format; Γιατί όχι; Έστω πως επιμέναμε να έχουμε την εντολή R-format "**`RR`**" με `op=001` και `funct=10`. Τότε, η εντολή "**`RR`** `x5`" με ποιάν άλλη εντολή I-format (και με τι τελεστέο) θα ήταν ίδια κι απαράλλακτη, με συνέπεια να μη μπορούμε να τις έχουμε και τις δύο στον RV_8; Αποδείξτε εν συντομία αλλά με "μαθηματική" ακρίβεια και σαφήνεια ότι το ίδιο θα ίσχυε γιά οιαδήποτε άλλη δυνατή εντολή R-format.

4(b): Εστω τώρα ότι "ξοδεύουμε" μόνο τους 7 από τους 8 διαθέσιμους συνδυασμούς του πεδίου op –τους 000, 001, 010, 011, 100, 101, και 110– γιά 7 εντολές I-format –τις **`ki, li, mi, ni, pi, qi, ri`**. Σ' αυτή την περίπτωση, πόσες εντολές R-format μπορούμε να έχουμε; Τι κώδικες **`op`** και **`funct`** θα έχει καθεμιά τους; Γιατί δεν μπορούμε να έχουμε περισσότερες από τόσες εντολές R-format;

4(c): Γιά την περίπτωση 4(b), σχεδιάστε ένα συνδυαστικό κύκλωμα, χρησιμοποιώντας πύλες NOT, AND, OR (οσωνδήποτε εισόδων) (μόνο τέτοιες, και όχι έτοιμους αποκωδικοποιητές), που να δέχεται σαν είσοδο μιαν εντολή (8 bits) και να παράγει σαν εξόδους:

- ένα σήμα I που ανάβει όταν και μόνον όταν η εντολή είναι I-format,
- ένα σήμα R που ανάβει όταν και μόνον όταν η εντολή είναι R-format,
- 7 σήματα, ένα γιά καθε εντολή I-format, που να ανάβει όταν και μόνον όταν βλέπει τη συγκεκριμένη εντολή, και
- ένα σήμα γιά καθε εντολή R-format, που να ανάβει όταν και μόνον όταν βλέπει τη συγκεκριμένη εντολή.

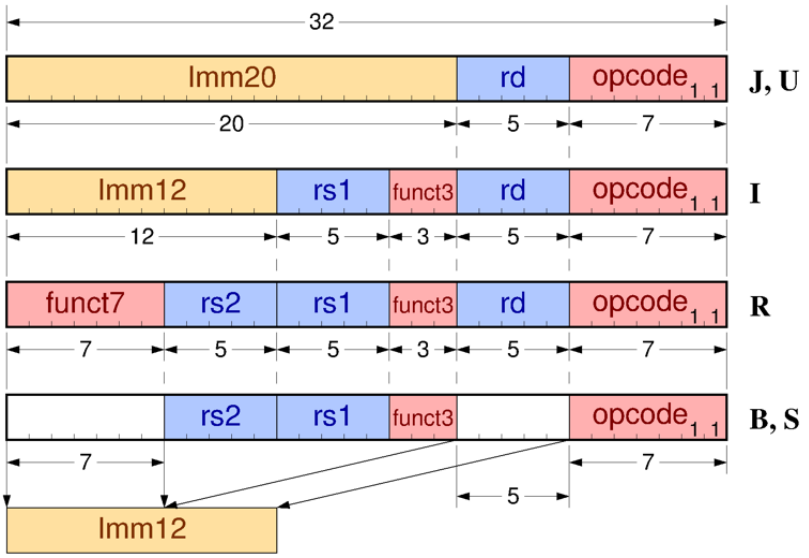
4(d): Ανάλογη ερώτηση με την 4(b), αλλά έστω ότι τώρα έχουμε μόνο 6 εντολές I-format. Πόσες εντολές R-format μπορούμε να έχουμε; Γράψτε τα opcodes (ή και function codes) όλων των εντολών, και των δύο format. Προσπαθήστε να επιλέξτε τα opcodes έτσι ώστε να απλοποιείται η αποκωδικοποίηση των σημάτων I και R (σύμφωνα με την εμπειρία σας από την ερώτηση 4(c), χωρίς πάντως να ζητείται εδώ σχεδίαση κυκλώματος).

4.3 Τα Format Εντολών του βασικού RISC-V:

Στο υπόλοιπο αυτού του μαθήματος θα ασχοληθούμε μόνο με τον βασικό RISC-V, του οποίου όλες οι εντολές είναι μεγέθους 32 bits. Μέσα σε αυτές τις εντολές όμως, άλλοτε χρειαζόμαστε 1, άλλοτε 2, και άλλοτε 3 ορίσματα καταχωρητών, και άλλες φορές χρειαζόμαστε σταθερή ποσότητα αντί καταχωρητή· έτσι, καταλήγει να χρειαζόμαστε διάφορες μορφές (format) εντολών, και με μεταβλητό μέγεθος (πλάτος) των πεδίων μέσα σε αυτές. Το σχήμα παρακάτω δείχνει τα βασικά format των εντολών του RISC-V. Πρώτον, όπως είπαμε παραπάνω, ο opcode βρίσκεται (ή τουλάχιστο ξεκινάει από) δεξιά (LS), και τα δύο δεξιά του bits είναι πάντα 11 αφού πρόκειται γιά 32-μπιτες εντολές.

Το πρώτο format του RISC-V στο σχήμα είναι το "**J**" ή "**U**"· πρόκειται στην πραγματικότητα γιά δύο ελαφρές παραλλαγές του ιδίου format, που διαφέρουν μεταξύ τους μόνο κατά την τοποθέτηση των bits εντός της σταθερής ποσότητας, όπως θα δούμε σε επόμενο κεφάλαιο, αλλά που αυτή τη στιγμή αρκεί να τις θεωρήσουμε σαν ένα και το αυτό format. Αυτό το format το ακολουθούν οι (τρείς μόνον) εντολές που χρειάζονται ένα (μόνον) όρισμα καταχωρητή, και μία φαρδιά –όσο μεγαλύτερη γίνεται– σταθερή ποσότητα. Έτσι, μετά τον opcode (πάντα από δεξιά προς τα αριστερά, σαν καλός Little Endian), υπάρχει ένα πεδίο καταχωρητή, και στη συνέχεια όλα τα υπόλοιπα bits της εντολής αφιερώνονται στη σταθερή ποσότητα. Ο καταχωρητής είναι καταχωρητής προορισμού γιά αυτές τις εντολές (δηλαδή αυτές γράφουν σε αυτόν), έτσι τον συμβολίζουμε "**rd**" (d = destination - προορισμός). Η σταθερή ποσότητα θέλουμε να έχει τόσα bits ώστε, αθροιζόμενα αυτά με τα bits των άλλων (μικρότερων) συνηθισμένων σταθερών του RISC-V, να μας δίνουν 32 bits, ούτως ώστε, όπως θα δούμε αργότερα, να μπορούμε να κατασκευάζουμε μιάν αυθαίρετη 32-μπιτη σταθερή ποσότητα μέσω δύο εντολών, μιάς εντολής U (**U**pper part of the 32-bit constant) όπως εδώ, και μιάς συνηθισμένης (π.χ. addi) εντολήςτου επόμενου, I-format. Οι σχεδιαστές του RISC-V θεώρησαν τα 12 bits σαν ικανοποιητικό μέγεθος γιά τις "συνηθισμένες" σταθερές ποσότητες, άρα χρειάζονταν 20 bits γιά τις "μεγάλες" σταθερές ποσότητες, εδώ στο U-format, επομένως ονοματίζουμε το σχετικό πεδίο **Imm20** στο σχήμα (20-bit Immediate constant). Αφού τα πεδία

καταχωρητών πρέπει να είναι 5 bits καθένα (2^5 = 32 κατσαχωρητές), περισσεύουν 7 bits γιά τον (πρώτο, βασικό) opcode, δεξιά –από τα οποία, βέβαια, τα δύο δεξιά είναι ήδη πιασμένα (δεσμευμένα να είναι 11) για την υποστήριξη των 16-μπιτων εντολών.



Το επόμενο βασικό format του RISC-V ονομάζεται "**I**", από τη λέξη **I**mmediate (constant), δηλαδή "σταθερή ποσότητα καθοριζόμενη *κατ' ευθείαν μέσα* στην εντολή. Αυτό φαίνεται δεύτερο στο σχήμα, και χρειάζεται δύο πεδία καταχωρητών, επιπλέον της σταθερής ποσότητας, όπως π.χ. οι εντολές addi και load που αποτελούν τυπικά παραδείγματα τέτοιων εντολών: ο ένας καταχωρητής είναι η πρώτη πηγή (source 1) πληροφορίας –άρα ονομάζεται "**rs1**"– η σταθερή ποσότητα είναι η δεύτερη πληροφορία πηγής - είσοδος στον υπολογισμό (ή την ανάγνωση μνήμης), και το αποτέλεσμα γράφεται στον καταχωρητή προορισμού, rd.

Οι σχεδιαστές του RISC-V χρειάζονταν αυξημένο χώρο opcode –περισσότερο απ' όσο άλλα ρεπερτόρια RISC– προκειμένου να αφήσουν χώρο γιά προαιρετικές επεκτάσεις, είτε πρότυπες είτε ιδιωτικές, γιά πρόβλεψη μελλοντικών επεκτάσεων του ρεπερτορίου, καθώς και γιά την κωδικοποίηση των 16-μπιτων εντολών. Είδαμε ότι αυτές οι τελευταίες "τρώνε" ήδη 2 από τα bits του βασικού opcode. Άρα, οι περισσότερες εντολές (πέρα από τις τρείς εντολές J/U-format) πρέπει να αφήνουν περισσότερα από 7 μόνο bits γιά τον opcode: πρόκειται ακριβώς γιά την ιδέα του opcode μεταβλητού μεγέθους (πλάτους) που είδαμε αμέσως προηγουμένως, στην άσκηση 4.2. Έτσι, το I-format, αντί να πιάνει και τα 20 bits του πεδίου Imm20 με 5 bits γιά τον rs1 και τα υπόλοιπα 15 bits γιά Immediate, *επεκτείνει* ουσιαστικά τον opcode κατά 3 ακόμα bits, τα οποία τα παίρνει από τη σταθερή ποσότητα, μικραίνοντάς την στα 12 bits. Αυτή την επέκταση του opcode κατά 3 bits την ονομάζουμε με το λίγο διαφορετικό όνομα *function code* γιά να συνενοούμαστε καθαρότερα, αλλά ουσιαστικά αποτελεί και αυτό το πεδίο, "**funct3**, κομάτι ή επέκταση του opcode. Το 12-μπιτο πεδίο που μένει, αριστερά, γιά τη σταθερή ποσότητα, το ονοματίζουμε επομένως "**Imm12**". *Ιστορικές σημειώσεις:* Οι RISC-I και RISC-II έδιναν 13 bits στις "συνηθισμένες" σταθερές, αντί 12 εδώ, και 19 bits στις "μεγάλες" σταθερές, αντί 20 εδώ, και ο opcode τους ήταν σταθερού μεγέθους 8 bits (από τα οποία το ένα αφιερώνονταν σαν "set condition codes flag" – κάτι του οποίου η μόδα έχει περάσει). Ο MIPS δίνει 16 bits γιά τις "συνηθισμένες" σταθερές, με αποτέλεσμα να του μένουν μόνον 6 bits γιά τον βασικό του opcode. Όλοι αυτοί οι υπολογιστές συνθέτουν μιάν αυθαίρετη 32-μπιτη σταθερή ποσότητα μέσω δύο εντολών που περιέχουν τα δύο κομάτια της σαν επιμέρους σταθερές: 20+12 bits στον RISC-V, 19+13 bits στους RISC-I και RISC-II, 16+16 bits στον MIPS.

Το άλλο από τα τρία βασικότερα format του RISC-V εικονίζεται στην τρίτη θέση του σχήματος, και ονομάζεται "**R**" format, επειδή έχει τρία πεδία καταχωρητών (**R**egister format). Σε σχέση με το I-format, εδώ αντικαθίσταται η σταθερή ποσότητα σαν δεύτερη πηγή πληροφορίας από τον δεύτερο καταχωρητή πηγής, "**rs2**" (source-2 register). Δεδομένου ότι τα πεδία καταχωρητών έχουν μόνον 5 bits, μας περισσεύουν τώρα και άλλα 7 bits ζαν τρίτη επέκταση του opcode, που την ονοματίζουμε, κατ' αναλογία όπως παραπάνω, "**funct7**".

Τελευταίο στο σχήμα φαίνεται άλλο ένα format του RISC-V, το οποίο αποτελεί παραλλαγή του I-format. (Όπως και με το J/U-format, έτσι κι εδώ υπάρχουν δύο παραλλαγές, η B και η S, που διαφέρουν μεταξύ τους μόνο κατά την τοποθέτηση των bits εντός της σταθερής ποσότητας, όπως θα δούμε σε επόμενο κεφάλαιο). Όπως και το I-format, έχουμε και εδώ δύο πεδία καταχωρητών και μία 12-μπιτη σταθερή ποσότητα. Η διαφορά είναι ότι εδώ οι δύο

καταχωρητές είναι και οι δύο πηγής, "**rs1**" και "**rs2**", αντί ένας πηγής και ένας προορισμού στο I-format. Οι εντολές store ακολουθούν το S-format: εδώ ο rs1 προστίθεται με το (signed) Imm12 γιά να δημιουργήσει τη διεύθυνση μνήμης, και τον rs2 τον διαβάζουμε σαν data γιά να αποσταλούν γιά εγγραφή στη μνήμη. Οι εντολές διακλάδωσης (branch) ακολουθούν το B-format: διαβάζουμε τους δύο καταχωρητές rs1 και rs2 γιά να τους συγκρίνουμε, η δε σταθερή ποσότητα χρησιμοποιείται γιά τον υπολογισμό της διεύθυνσης προορισμού της διακλάδωσης. Το B/S-format φαίνεται παράξενο επειδή η σταθερή ποσότητα Imm12 είναι "σπασμένη" σε δύο κομάτια μέσα στην εντολή· αυτό γίνεται γιά τον εξής λόγο:

Ο RISC-V έχει το περισσότερο εξελιγμένο format εντολών όσον αφορά τη **σταθερή θέση** των τελεστέων-καταχωρητών (register operands) μέσα στην εντολή. Ο ένας λόγος που θέλουμε τα πεδία των καταχωρητών *πηγής* σε σταθερές θέσεις μέσα στην εντολή, ανεξαρτήτως format, είναι γιά να ξέρουμε πάντα ποιοί είναι αυτοί (εάν υπάρχουν, ο ένας ή και οι δύο) **πριν** αποκωδικοποιήσουμε τον opcode γιά να μάθουμε ποιό είναι το format της, και **χωρίς** την ανάγκη πολυπλέκτη στο δρόμο προς την είσοδο διευθύνσεων του αρχείου των καταχωρητών, ούτως ώστε να αρχίσει η ανάγνωση αυτών των καταχωρητών πηγής από εκεί αμέσως μόλις έλθει η εντολή στον επεξεργαστή από τη μνήμη εντολών, χωρίς ενδιάμεση καθυστέρηση και *εν παραλλήλω* με την αποκωδικοποίηση του opcode. Το χαρακτηριστικό αυτό, των καταχωρητών *πηγής* σε σταθερές θέσεις μέσα στην εντολή, ανεξαρτήτως format, το είχε και ο MIPS· όμως στον MIPS, ο καταχωρητής προορισμού είναι σε μεταβλητή θέση μέσα στο format, με το σκεπτικό ότι τον καταχωρητή προορισμού τον χρειαζόμαστε να τον ξέρουμε *αργότερα* στη διάρκεια εκτέλεσης της εντολής, όταν θα είναι έτοιμο το αποτελεσμα γιά να γραφτεί εκεί, και άρα μέχρι τότε υπάρχει ο χρόνος να έχει αποκωδικοποιηθεί ο opcode και να έχει επιλεγεί το σχετικό πεδίο μέσα από την εντολή.

Ο RISC-V είναι πιό εξελιγμένος, και κρατά **και** τον καταχωρητή προορισμού σε σταθερή θέση, γιά έναν πρόσθετο λόγο. Στους επεξεργαστές "Superscalar" (§10.4, §14), διαβάζουμε *δύο ή περισσότερες εντολές ταυτόχορνα* από τη μνήμη εντολών, και ελέγχουμε εάν η επόμενη(ες) *δεν εξαρτάται* από την προηγούμενη, δηλαδή δεν χρησιμοποιεί το αποτέλεσμά της, οπότε όταν είναι έτσι, ανεξάρτητη, μπορούμε να την εκτελέσουμε *ταυτόχρονα* με την προηγούμενη, χρησιμοποιώντας πρόσθετες πόρτες του αρχείου καταχωρητών και πρόσθετες αριθμητικές μονάδες (ALU's). Γιά να ελέγξουμε την εξάρτηση ή ανεξαρτησία δύο εντολών, συγκρίνουμε τους αριθμούς (διευθύνσεις) των καταχωρητών πηγής της επόμενης με τον αριθμό του καταχωρητή προορισμού της προηγούμενης. Γιά να μπορούμε να κάνουμε αυτό τον έλεγχο γρήγορα, πριν την αποκωδικοποίηση των opcodes και χωρίς την ανάγκη πολυπλεκτών επιλογής από διάφορα πεδία μέσα στις εντολές, θέλουμε και οι καταχωρητές πηγής αλλά **και** ο καταχωρητής προορισμού να είναι σε σταθερές, πάντα τις ίδιες θέσεις μέσα στην εντολή. Γιά να μπορέσει να ικανοποιηθεί αυτή η απαίτηση, να είναι πάντα στις ίδιες θέσεις και οι καταχωρητές πηγής και ο προορισμού, ο RISC-V αναγκάστηκε, όπως φάινεται στο σχήμα, να "σπάσει" το πεδίο Imm12 στο B/S-format, σε δύο κομάτια –τα 7 bits αριστερά, και τα 5 bits δεξιότερα. Δεδομένου ότι το πεδίο Imm12 το χρειαζόμαστε συνήθως αργότερα στη διάρκεια της εκτέλεσης της εντολής (μετά την ανάγνωση του καταχρητή rs1 από το αρχείο καταχωρητών, γιά να γίνει αριθμητική πράξη μεταξύ της τιμής του rs1 και του Imm12), υπάρχει εν τω μεταξύ χρόνος να έχει αποκωδικοποιηθεί ο opcode και να έχει λειτουργήσει και ο απαιτούμενος πολυπλέκτης.

Τις πλήρεις λεπτομέρειες γιά τα formats των εντολών του RISC-V θα τις βρείτε στις σελίδες 15-17 του εγχειρίδιου (manual) του RISC-V που λέγαμε νωρίτερα: riscv-spec-20191213.pdf (από τη σελίδα riscv.org/specifications/). Γιά περισσότερες πληροφορίες, δείτε και τις σελίδες 129-130 (ή και εώς 136) του ίδιου εγχειριδίου, και γιά τον πλήρη ορισμό του βασικού ρεπερτορίου εντολών τις σελίδες 17-29.

Τρόπος Παράδοσης:
Κάντε την απλή αυτή άσκηση από τώρα, γιά να την έχετε έτοιμη, παρ' ότι θα την παραδώσετε λίγο αργότερα, μαζί με την επόμενη σειρά ασκήσεων 5. Ο τρόπος παράδοσης, γιά τις ασκήσεις 4 και 5 μαζί, θα είναι on-line, σε μορφή PDF (μπορεί να είναι κείμενο μηχανογραφημένο ή/και "σκαναρισμένο" χειρόγραφο).
