Ιόνιο Πανεπιστήμιο – Τμήμα Πληροφορικής

Αρχιτεκτονική Υπολογιστών

Σχηματισμός διευθύνσεων μνήμης

(Memory addressing modes)

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

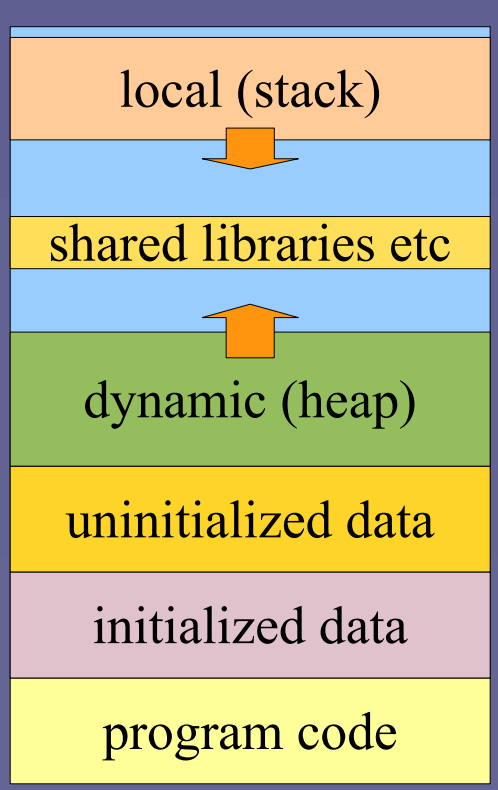
Μ.Στεφανιδάκης

Χρήση Διευθύνσεων Μνήμης

- Προσπέλαση δεδομένων (μεταβλητές προγραμμάτων)
  - Εντολές μηχανής που διαβάζουν ή γράφουν δεδομένα σε διευθύνσεις μνήμης
  - Μόνο οι εντολές load και store σε μια τυπική αρχιτεκτονική RISC
  - Πολύ περισσότερες εντολές σε αρχιτεκτονικές CISC
- Ανάκληση εντολών (ροή εκτέλεσης)
  - Ποια θα είναι η επόμενη προς εκτέλεση εντολή
  - Οι εντολές διακλάδωσης (branches/jumps) περιγράφουν μια διεύθυνση στη μνήμη όπου θα μεταφερθεί η ροή εκτέλεσης αν ισχύει κάποια συνθήκη

Χώρος Διευθύνσεων Μνήμης

- Κάθε εκτελούμενο πρόγραμμα «βλέπει» μια περιοχή διευθύνσεων μνήμης όπου τοποθετούνται ο κώδικας και τα δεδομένα του



Δεν πρέπει να ξεχνάμε ότι στα συστήματα με εικονική μνήμη (virtual memory) πρόκειται για χώρο λογικών διευθύνσεων κι όχι φυσικών!

Διευθύνσεις δεδομένων

- Θα πρέπει να υποστηρίζονται όλα τα «κλασικά» χαρακτηριστικά των δομημένων γλωσσών προγραμματισμού
  - Στατικά (static/global) δεδομένα, βρίσκονται σε σταθερή θέση σε όλη τη διάρκεια «ζωής» του προγράμματος
  - Τοπικά (local) δεδομένα, σε κλήσεις συναρτήσεων και nested μπλοκ κώδικα
  - Οργάνωση σε πίνακες όμοιων στοιχείων (arrays) και δομές (structures)

Απόλυτη διεύθυνση μνήμης

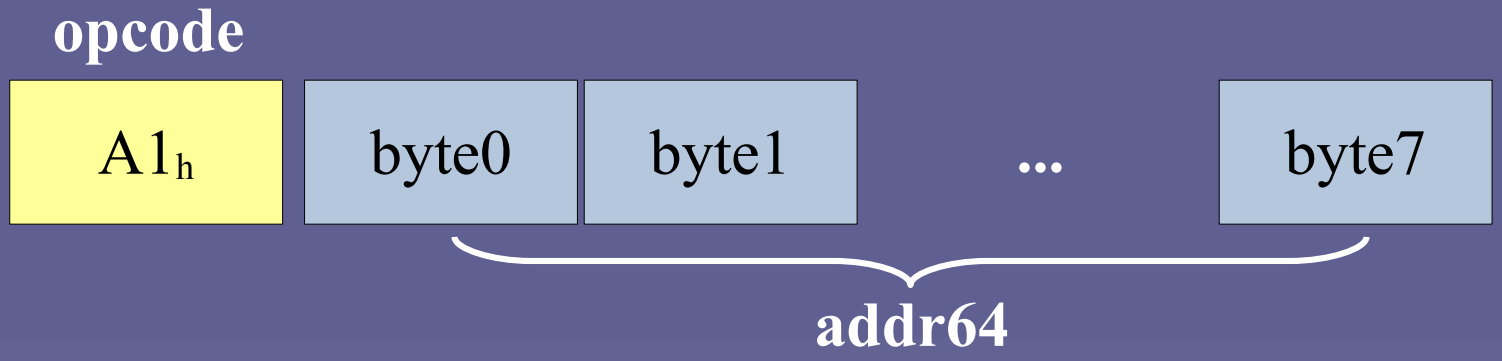
- Direct/absolute addressing
  - Η διεύθυνση μνήμης για ανάγνωση ή εγγραφή βρίσκεται ως δυαδικός αριθμός χωρίς πρόσημο μέσα στα bits της εντολής
  - Μπορεί να χρησιμοποιηθεί μόνο όταν η θέση μιας μεταβλητής είναι γνωστή πριν την εκτέλεση του προγράμματος
    - Για να μπορεί να ενσωματωθεί στον κώδικα κατά τη μεταγλώττιση (assembler/linker) ή κατά τη φόρτωση (loader) του προγράμματος
    - Στατικές/global μεταβλητές
  - Στην πράξη η μορφή αυτή σχηματισμού διεύθυνσης χρησιμοποιείται πολύ σπάνια
    - Ακόμα και για στατικές μεταβλητές μπορούν να χρησιμοποιηθούν άλλοι τρόποι αν το πρόγραμμα έχει μικρό μέγεθος (έως 4GB)

Η αρχιτεκτονική x86-64 (64-bit mode)

- CISC αρχιτεκτονική συνόλου εντολών
  - Διευθύνσεις μνήμης (πηγή ή προορισμός) μπορούν να εμφανιστούν και σε άλλες εντολές εκτός από εντολές τύπου load – store (π.χ. add).
  - Μεταβλητού μήκους εντολές μηχανής (1-15 bytes)
  - Default: 64-bit διευθύνσεις, 32-bit δεδομένα
    - Αλλαγή (π.χ. 64-bit δεδομένα) μέσω prefix bytes
  - 16 64-bit καταχωρητές γενικού σκοπού
    - RAX, RBX, RCX, RDX, RSI, RDI, RBP, RSP, R8 – R15
    - Ορισμένες εντολές χρησιμοποιούν συγκεκριμένους καταχωρητές (μη ομοιόμορφο σετ εντολών)
    - Μπορεί να χρησιμοποιηθεί το 32-bit, 16-bit, 8-bit μέρος ενός καταχωρητή, π.χ. RAX (64 bits)/EAX (32-bits)/AX (16-bits)/AL (8-bits)

x86-64 (64-bit mode): εντολές με απόλυτη διεύθυνση

- Παράδειγμα εντολής

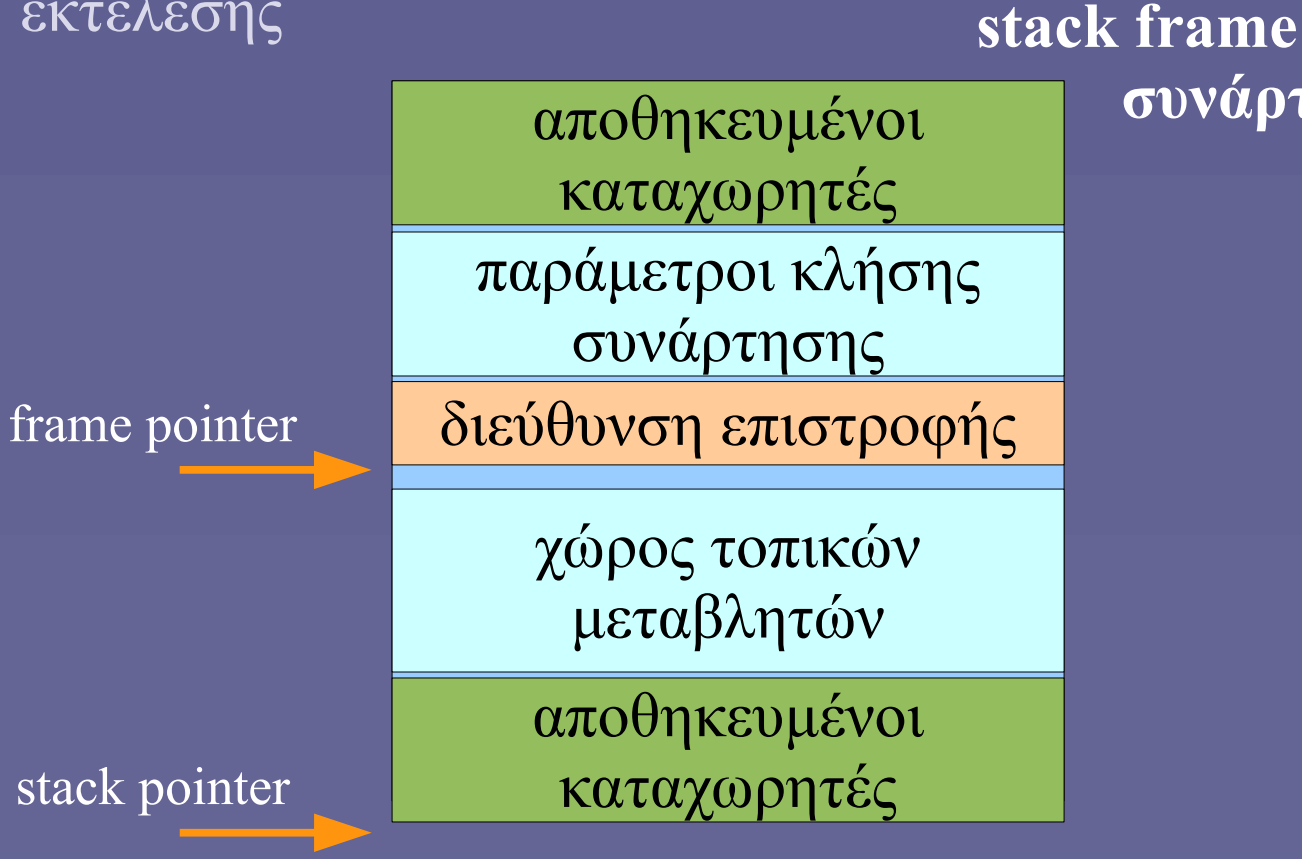


eax ← mem[addr64]

- Μόνο ο καταχωρητής RAX/EAX/AX/AL ως πηγή ή προορισμός
  - Με διάφορους συνδυασμούς prefix & opcode ρυθμίζεται το εύρος των μεταφερόμενων δεδομένων (64, 32, 16 ή 8 bits)

Τοπικές μεταβλητές

- Η θέση τους δεν είναι γνωστή πριν την εκτέλεση
  - Τοποθετούνται στο stack frame των κλήσεων των συναρτήσεων (ή των nested μπλοκ κώδικα)
  - Η θέση κάθε stack frame στη μνήμη εξαρτάται από τη ροή εκτέλεσης



Frame pointer

- Κατά τη μεταγλώττιση δεν ξέρουμε πού θα βρεθεί κάθε stack frame
  - Συνεπώς δεν μπορούμε να χρησιμοποιήσουμε απόλυτες διευθύνσεις μνήμης
- Ξέρουμε όμως κάθε τοπική μεταβλητή πού βρίσκεται μέσα στο stack frame
  - Σχετικά ως προς την αρχή του stack frame
- Frame pointer (fp)
  - Καταχωρητής (συνήθως γενικού σκοπού) που περιέχει τη διεύθυνση της αρχής του stack frame
  - Ενημερώνεται σε κάθε κλήση/επιστροφή συνάρτησης
  - Όλες οι προσπελάσεις σε τοπικές μεταβλητές γίνονται σε σχέση (±) με την τρέχουσα τιμή του fp

Σχηματισμός διεύθυνσης για τοπικές μεταβλητές

- Η διεύθυνση ανάγνωσης/εγγραφής θα πρέπει να σχηματίζεται συνδυάζοντας
  - Την τιμή ενός καταχωρητή, όπως ο frame pointer (base)
  - Τη σχετική απόσταση της τοπικής μεταβλητής από τον fp (offset)

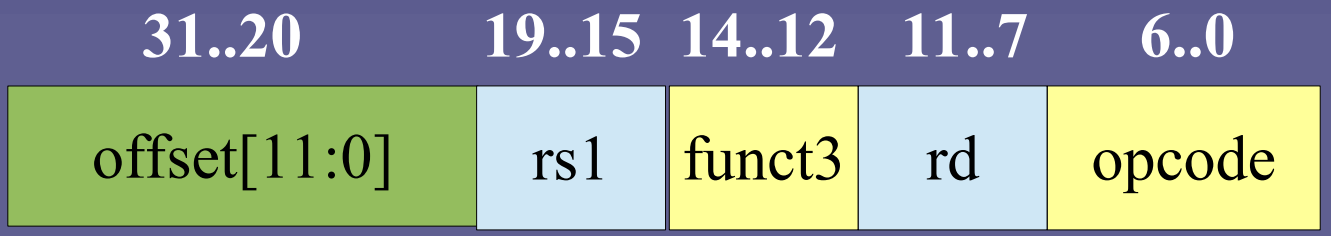
$$\text{διεύθυνση} = [\text{base-reg}] + \text{offset}$$

- Displacement addressing mode
  - Συνδυασμός τιμής καταχωρητή (base) και μικρής σταθεράς με πρόσημο (offset)
    - Το offset διαθέτει λιγότερα bits από το πλήρες εύρος διεύθυνσης (32 ή 64 bits)

Η βασική αρχιτεκτονική RISC-V (32- και 64-bit)

- Περιγράφει ένα μικρό σύνολο βασικών εντολών
  - Επιπλέον εντολές περιγράφονται ως επεκτάσεις (extensions)
  - Είναι κλασική load – store RISC αρχιτεκτονική
    - Μόνο εντολές load/store (ανάγνωση/εγγραφή μνήμης) μπορούν να χρησιμοποιήσουν διεύθυνση μνήμης
- Διαθέτει 32 καταχωρητές γενικού σκοπού
  - x0 – x31
  - Μπορούν να χρησιμοποιηθούν ομοιόμορφα από τις εντολές
  - Κατά σύμβαση (δεν επιβάλλεται από την αρχιτεκτονική) κάποιοι καταχωρητές χρησιμοποιούνται για συγκεκριμένο σκοπό (π.χ. ο x8 ως frame pointer)
- Ο σχηματισμός διευθύνσεων [base-reg] + offset είναι ο μοναδικός που υποστηρίζεται

Load (I-type) στα 32 bits



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

εύρος μεταφοράς

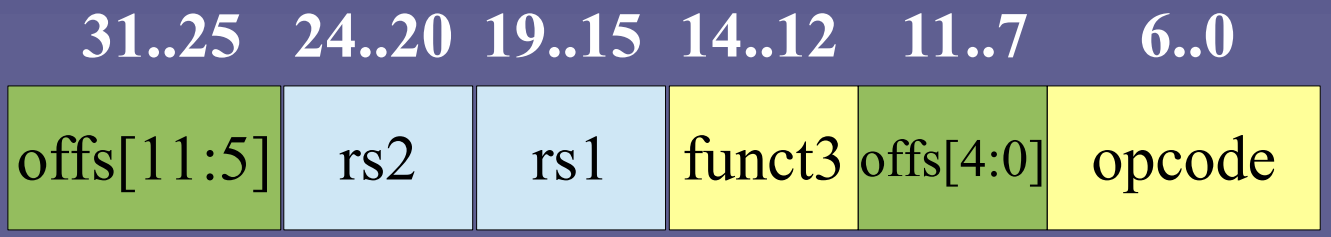
0000011

rd ← mem[rs1 ± offset]

base-reg

12 bits με πρόσημο (±2KB)

Store (S-type) στα 32 bits



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

εύρος μεταφοράς

0100011

rs2 → mem[rs1 ± offset]

Προσπέλαση μέσω δεικτών

- Η χρήση δεικτών είναι ένα τυπικό παράδειγμα δυναμικής διεύθυνσης
  - Το περιεχόμενο του δείκτη (διεύθυνση μνήμης) είναι γνωστό μόνο κατά την εκτέλεση

val = *ptr;

- Register indirect addressing mode
  - Το περιεχόμενο ενός καταχωρητή (reg) χρησιμοποιείται για την παραγωγή της διεύθυνσης μνήμης

διεύθυνση = [reg]

  - Μπορεί να θεωρηθεί ως υποπερίπτωση του [reg] + offset έχοντας offset = 0

Arrays

- Για την ανάγνωση/εγγραφή του i-οστού στοιχείου

$$val = a[i];$$

  - Θα πρέπει να ξέρουμε την αρχή του array (&a[0])
  - Την τιμή του i
  - Το μέγεθος κάθε στοιχείου του array
- Indexed addressing mode
  - Ένας καταχωρητής (reg1) περιέχει τη βάση (αρχή) του array και ένας δεύτερος (reg2) την απόσταση του στοιχείου i από την αρχή του array

$$\text{διεύθυνση} = [reg1] + [reg2]$$

Scaled Index

- Στις πιο κοινές περιπτώσεις τα arrays περιέχουν char, int, double κ.ο.κ στοιχεία
  - Με αντίστοιχο μέγεθος d ανά στοιχείο (1, 4, 8 bytes)
  - Η θέση του i-οστού στοιχείου υπολογίζεται ως

$$\text{αρχή array} + i * d$$

- Scaled - Indexed addressing mode
  - Ορισμένες αρχιτεκτονικές υποστηρίζουν τον πιο πάνω υπολογισμό για κοινά d (1, 2, 4, 8)

$$\text{διεύθυνση} = [\text{reg1}] + [\text{reg2}*d]$$

  - Σε παραλλαγή του προηγούμενου, μπορεί να προστεθεί και ένα επιπλέον offset στη διεύθυνση