

Ιόνιο Πανεπιστήμιο – Τμήμα Πληροφορικής
Αρχιτεκτονική Υπολογιστών
2025-26

Αρχιτεκτονικές Συνόλου Εντολών

(Instruction Set Architectures - ISA)

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

Μ.Στεφανιδάκης



Ο (μικρο)επεξεργαστής

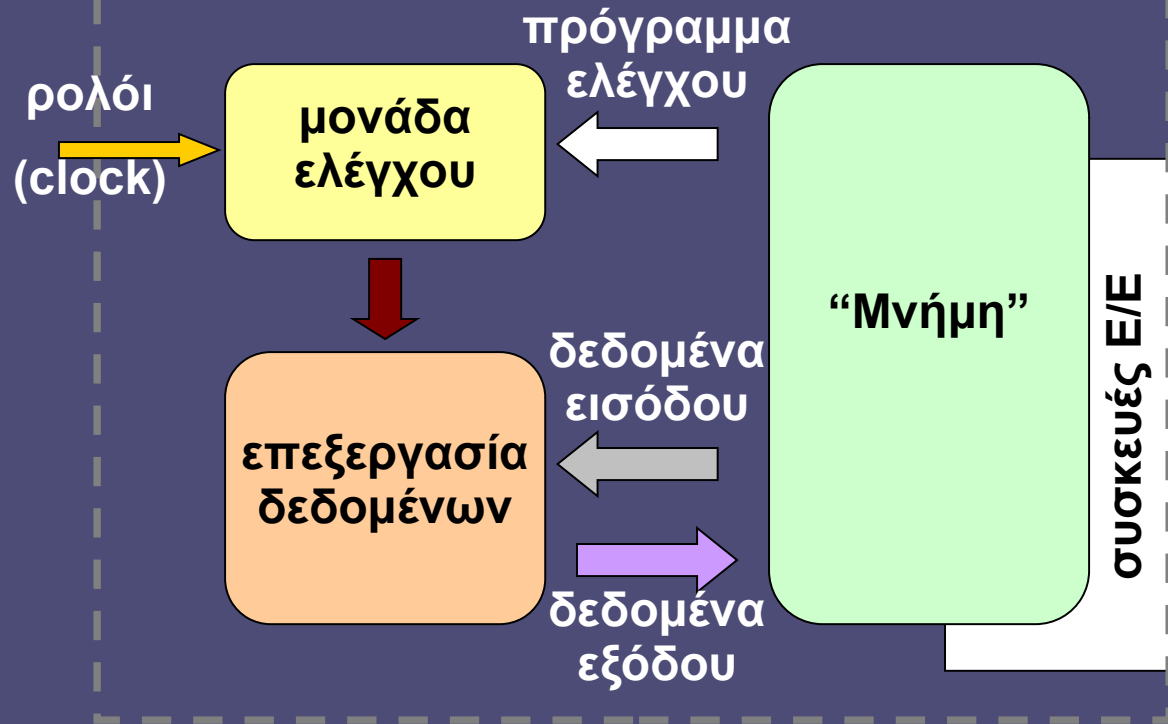
- (Micro)processor
 - Αρχικά, μόνο η Κεντρική Μονάδα Επεξεργασίας (ΚΜΕ)
 - Central Processing Unit (CPU)
 - Περιέχει σήμερα πολλαπλές υπομονάδες επεξεργασίας
 - Σε κάθε «πυρήνα» (core)
 - Με διαφορετικά χαρακτηριστικά ή/και ρόλους
 - Και μέρος της ιεραρχίας μνήμης (κρυφές μνήμες)
 - Καθώς και μέρος των διεπαφών (interfaces) επικοινωνίας με την κύρια μνήμη και τις μονάδες εισόδου - εξόδου

Κεντρική Μονάδα Επεξεργασίας

- Central Processing Unit (CPU)
 - Ένας όρος που τείνει προς εξαφάνιση
 - Μετά την εμφάνιση των πολλών/διαφορετικών μονάδων επεξεργασίας στο ίδιο τσιπ
- Ποιος ο ρόλος μιας Μονάδας Επεξεργασίας
 - Μετασχηματίζει (επεξεργάζεται) **δεδομένα** σύμφωνα με ένα **πρόγραμμα ελέγχου**
 - Το πρόγραμμα ελέγχου αποτελείται από **εντολές μηχανής**

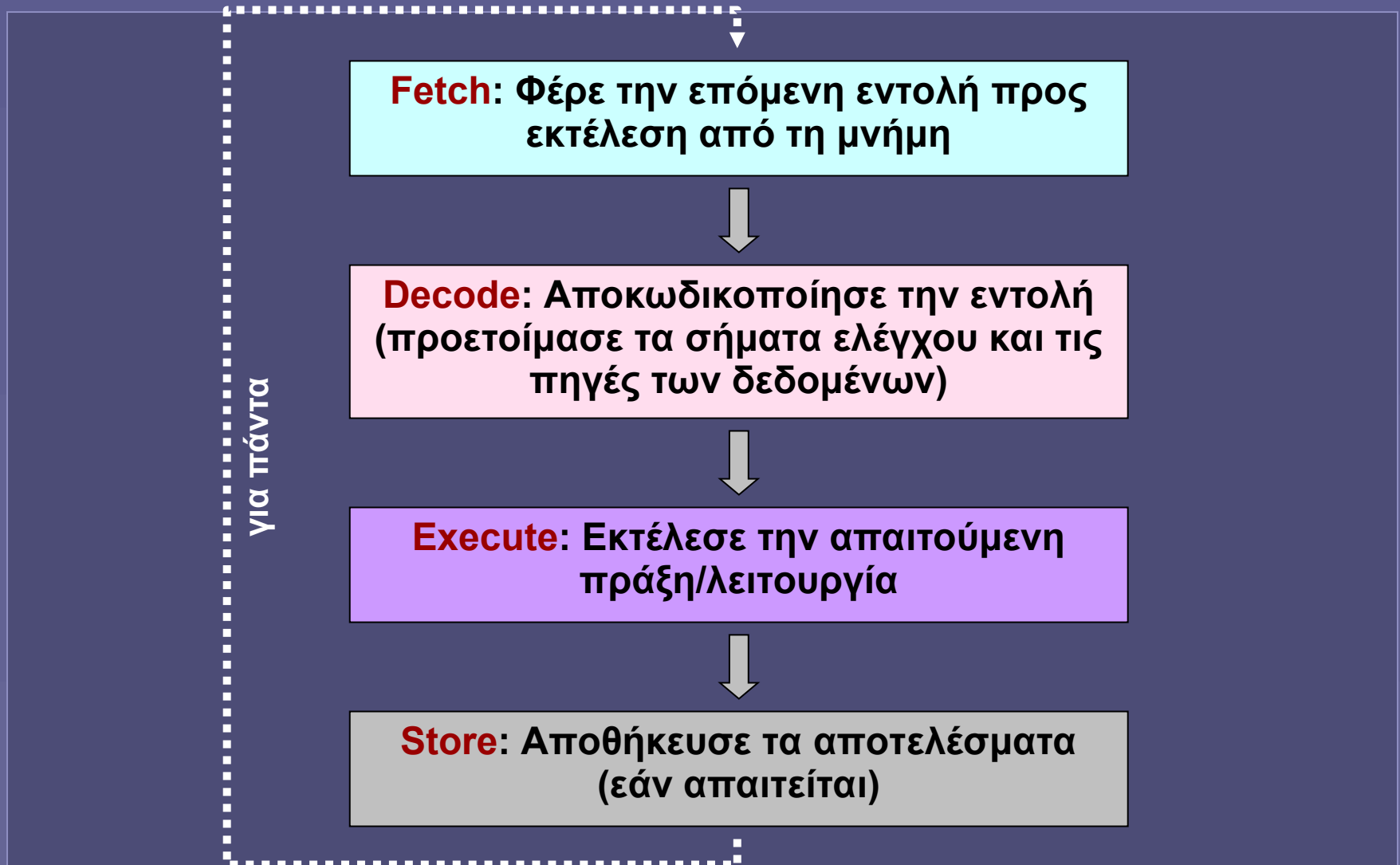
Το μοντέλο «von Neumann»

υπολογιστικό σύστημα



- Το **πρόγραμμα ελέγχου**, όπως και **τα δεδομένα**, αποθηκεύονται στη **μνήμη** του υπολογιστή
 - “Stored-program computer”

Εκτέλεση εντολών: ο κύκλος μηχανής



Εκτέλεση εντολών

- Program Counter (PC) ή Instruction Pointer (IP)
 - Καταχωρητής ειδικού σκοπού
 - Περιέχει τη διεύθυνση της θέσης μνήμης όπου βρίσκεται η εντολή προς εκτέλεση
- Αύξηση PC μετά την ανάκληση της εντολής ($PC += d$)
 - d = αριθμός bytes εντολής που ανακλήθηκε
 - Μετάβαση στην επόμενη εντολή
- Εξαίρεση: εντολές διακλάδωσης
 - Μεταπήδηση σε νέα θέση μνήμης (ο PC παίρνει νέα τιμή)
- Η διαδικασία επαναλαμβάνεται συνεχώς
 - Όσο η Μονάδα Επεξεργασίας είναι σε λειτουργία

Η πρώτη εντολή που εκτελείται

- Εκκίνηση εκτέλεσης
 - Με την εφαρμογή τάσης ο PC παίρνει μια προκαθορισμένη τιμή
 - Συνήθως στην αρχή ή στο τέλος της υποστηριζόμενης περιοχής μνήμης
 - Ανάλογα με την αρχιτεκτονική της κάθε Μονάδας Επεξεργασίας
 - Εκεί ο κατασκευαστής έχει τοποθετήσει τις πρώτες εντολές αρχικοποίησης του συστήματος

Αρχιτεκτονική Συνόλου Εντολών

- **Instruction Set Architecture (ISA)**
 - Το ορατό μέρος ενός υπολογιστικού συστήματος για τον προγραμματιστή (και τον μεταγλωττιστή)
 - Δεκαετία 60-70: συνώνυμο του όρου «**αρχιτεκτονική H/Y**»
 - «η δομή ενός υπολογιστή, την οποία ο προγραμματιστής πρέπει να γνωρίζει για να γράψει ένα σωστό (**χρονικά ανεξάρτητο**) πρόγραμμα σε γλώσσα μηχανής για τον υπολογιστή αυτόν» (IBM)
 - Σήμερα χρησιμοποιείται και ο όρος «**μακρο-αρχιτεκτονική**» με την ίδια σημασία

Η διεπαφή ISA στην ιεραρχία επιπέδων



- Αρχιτεκτονική Εντολών (ISA)
 - Η διεπαφή υλικού-λογισμικού

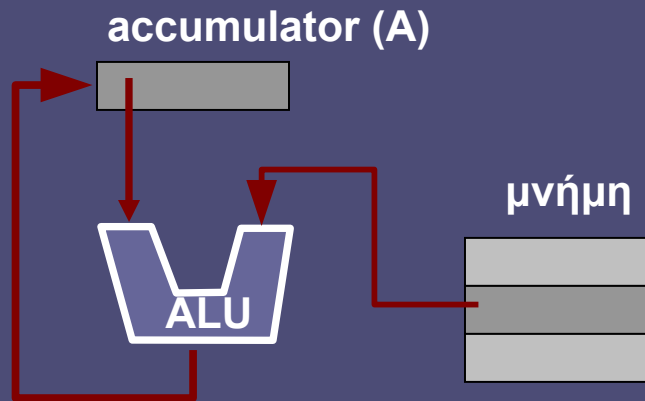
Αρχιτεκτονική Συνόλου Εντολών (ISA)

- Τι περιγράφει;
 - Διαθέσιμες πράξεις/λειτουργίες
 - Κωδικοποίηση λειτουργιών
 - Μορφή των δεδομένων εισόδου-εξόδου
 - Operands
 - Μέθοδοι προσπέλασης μνήμης
 - Προέλευση των δεδομένων, σχηματισμός διευθύνσεων μνήμης
 - Χώροι προσωρινής αποθήκευσης
 - Διαθέσιμοι καταχωρητές στον προγραμματιστή
 - Διακοπές και καταστάσεις σφάλματος
 - Ποια η “αντίδραση” του επεξεργαστή

Εξέλιξη των αρχιτεκτονικών συνόλου εντολών

- Οι πρώτοι υπολογιστές (.. - '60)
 - Αρχιτεκτονικές συσσωρευτή (accumulator)
 - Και αργότερα στοίβας (stack)
 - Ικανοποιητική λύση λόγω της τεχνολογίας και των μεταγλωττιστών της εποχής εκείνης

Αρχιτεκτονική συσσωρευτή (accumulator)



π.χ. η εντολή:

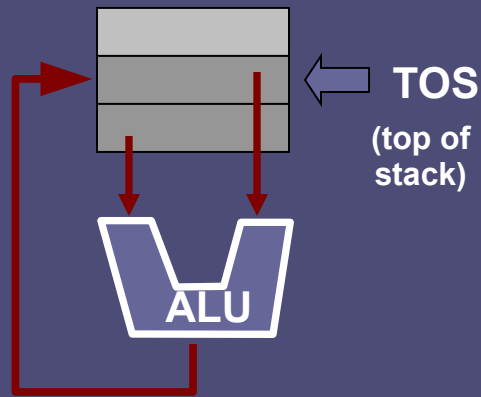
add X

σημαίνει:

$A \leftarrow A + X$

- Μια πηγή (είσοδος) δεδομένων και ταυτόχρονα η θέση αποθήκευσης του αποτελέσματος είναι πάντα ο **συσσωρευτής**
- **1-address architecture**, τα bits της εντολής περιγράφουν μόνο μία (τη δεύτερη) πηγή δεδομένων
- **Αρχιτεκτονική των πρώτων υπολογιστών**

Αρχιτεκτονική στοίβας (stack)



π.χ. η εντολή:

add

σημαίνει:

$t \leftarrow \text{pop}()$

$u \leftarrow \text{pop}()$

$v \leftarrow t + u$

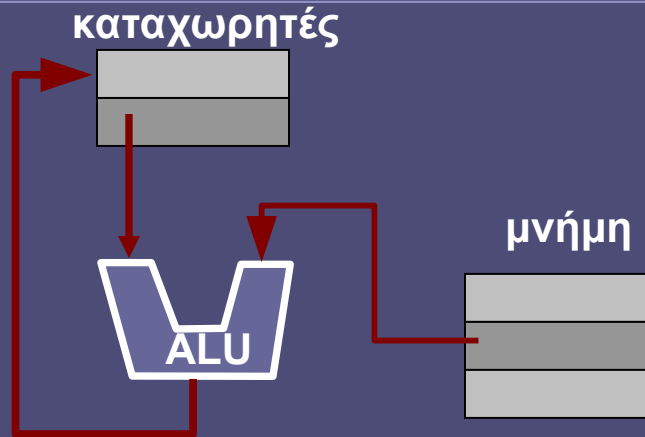
push(v)

- Οι πηγές προσδιορίζονται **έμμεσα**
 - Βρίσκονται στην κορυφή της στοίβας
 - **0-address architecture**, τα bits της εντολής δεν περιγράφουν καμία πηγή δεδομένων
 - Δημοφιλές σχήμα κατά τη δεκαετία του 60

Εξέλιξη των αρχιτεκτονικών συνόλου εντολών (2)

- **Πολύπλοκες αρχιτεκτονικές ('70 - ..)**
 - Ενσωμάτωση σύνθετων μορφών εντολών και μεθόδων προσπέλασης μνήμης
 - Προσπάθεια υποστήριξης υψηλών γλωσσών προγραμματισμού – μείωσης κόστους λογισμικού
 - Οι μεταγλωττιστές δεν είχαν φτάσει ακόμα σε υψηλό επίπεδο
 - Πολλά χαρακτηριστικά έμεναν όμως αχρησιμοποίητα...
 - **Complex Instruction Set Computers (CISC)**

Αρχιτεκτονικές με καταχωρητές (registers)



π.χ. η εντολή:

add R2, R1, mem(100)

σημαίνει:

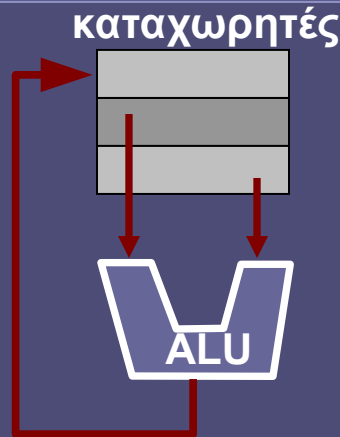
$R2 \leftarrow R1 + \text{mem}[100]$

- **Memory-register**
 - Οποιαδήποτε εντολή μπορεί να προσπελάσει τη μνήμη
- Πολλαπλές προσπελάσεις μνήμης ανά εντολή
 - Λήψη εντολής – Λήψη δεδομένων εντολής
 - Συνωστισμός στον δίαυλο επικοινωνίας με μνήμη
- Πολύπλοκη εκτέλεση εντολής σε πολλαπλά στάδια

Η εξέλιξη της αρχιτεκτονικής εντολών (3)

- **Reduced Instruction Set Computers (RISC) ('80 - ...)**
 - Απλούστερες και φθηνότερες load-store αρχιτεκτονικές με σταθερό μήκος εντολών
 - Μεγαλύτερη απόδοση – ταχύτερη εκτέλεση εντολών
 - Ευνοείται από την αφθονία υλικού χαμηλού κόστους και την προηγμένη τεχνολογία των μεταγλωττιστών

Αρχιτεκτονικές με καταχωρητές (registers)



π.χ. η εντολή:

add R1, R2,R3

σημαίνει:

$R1 \leftarrow R2 + R3$

- **Register-register (load-store)**
 - Μόνο ξεχωριστές εντολές **load** (ανάγνωση) – **store** (εγγραφή) μπορούν να προσπελάσουν τη μνήμη
- **Η αρχιτεκτονική των σύγχρονων επεξεργαστών**
 - Οι καταχωρητές προσπελούνται πολύ γρήγορα
 - Η εκτέλεση των εντολών απαιτεί λιγότερα στάδια
 - Χρειάζονται λιγότερα bits για την επιλογή καταχωρητών (από τις διευθύνσεις μνήμης)

Τα όρια μεταξύ CISC – RISC

- Σήμερα δεν είναι πάντα σαφής ο διαχωρισμός
 - Οι σημερινοί επεξεργαστές με εντολές CISC (π.χ. η αρχιτεκτονική x86), μεταφράζουν εσωτερικά μέσω πρόσθετου υλικού (hardware) σε μικροεντολές (μOPs)
 - Αντίστοιχα, επεξεργαστές RISC μπορούν κατά την εκτέλεση να συνδυάζουν απλές εντολές σε πιο σύνθετες λειτουργίες (fusion)
 - Είναι θέμα «μικρο-αρχιτεκτονικής» (υλοποίησης) κι όχι «μακρο-αρχιτεκτονικής» (ISA)
 - Ένα τυπικό κριτήριο για το αν ένας επεξεργαστής έχει CISC ή RISC ISA είναι αν μόνο εντολές load – store επιτρέπουν προσπέλαση στη μνήμη (RISC) ή αν κάθε είδους εντολή μπορεί να προσπελάσει τη μνήμη (CISC)

Κωδικοποίηση Εντολών

- Κάθε εντολή είναι μια σειρά δυαδικών ψηφίων



Περιγράφει το είδος της πράξης που θα εκτελεστεί

Περιγράφουν την **προέλευση** των δεδομένων εισόδου (αριθμό καταχωρητή, διεύθυνση μνήμης κλπ) και τον **προορισμό** των δεδομένων εξόδου (αποτελέσματος πράξης)

Το είδος της πράξης (opcode) προσδιορίζει τον τύπο, την προέλευση και τον αριθμό των δεδομένων που συμμετέχουν στην πράξη

Μεταβλητού ή σταθερού μήκους;

- Διαφορετικές αρχιτεκτονικές συνόλου εντολών
 - **Μεταβλητού μήκους:** οι εντολές δεν έχουν τον ίδιο αριθμό bits
 - Συμπαγή προγράμματα (οι συχνά χρησιμοποιούμενες εντολές έχουν μικρότερο μέγεθος)
 - Σημαντικά πολυπλοκότερο υλικό αποκωδικοποίησης
 - **Σταθερού μήκους:** όλες οι εντολές έχουν τον ίδιο αριθμό bits (π.χ. 32 bits)
 - Μεγαλύτερα προγράμματα (οι εντολές έχουν μεγαλύτερο μέγεθος)
 - Απλούστερη και ταχύτερη λήψη-αποκωδικοποίηση

Εντολές: κατηγορίες λειτουργιών

- Οι τρεις βασικές κατηγορίες
 - Αριθμητικές και λογικές πράξεις
 - Από δύο πηγές εισόδου (καταχωρητές/μνήμη) προς έναν προορισμό (καταχωρητή/μνήμη)
 - Μεταφορά δεδομένων
 - Από-πρός καταχωρητές/μνήμη
 - Έλεγχος ροής εκτέλεσης
 - Διακλαδώσεις και κλήσεις/επιστροφές συναρτήσεων
 - Ειδικές εντολές διακοπής εκτέλεσης

Αριθμητικές/λογικές εντολές

- Αριθμητικές-λογικές πράξεις

- Είδος πράξης
- Είδος δεδομένων
- Πηγές δεδομένων και προορισμός
- Παράδειγμα (θεωρητικό):
 - $Rd = Rs1 + Rs2$

add	Rd	Rs1	Rs2
-----	----	-----	-----

- Rd: προορισμός (καταχωρητής αποθήκευσης αποτελέσματος)
- Rs1, Rs2: πηγές (καταχωρητές δεδομένων εισόδου)

Εντολές μεταφοράς δεδομένων

- Μεταφορά δεδομένων
 - Πηγή δεδομένων και προορισμός
 - Το μήκος της μεταφερόμενης λέξης
 - 64, 32, 16 ή 8 bits
 - Παράδειγμα (θεωρητικό):
 - $Rd = \text{mem}[\text{addr}]$



- Στο παράδειγμα η διεύθυνση είναι **απόλυτη** (προσδιορίζεται μέσα στην εντολή)
 - Δεν είναι η χρησιμότερη μορφή διεύθυνσης!

Μέθοδοι προσπέλασης μνήμης

- Ορισμένες εντολές προσπελαύνουν τη μνήμη για ανάγνωση ή εγγραφή δεδομένων
 - Σχεδιασμός για υποβοήθηση του λογισμικού
 - Τοπικές μεταβλητές
 - Δείκτες (έμμεση προσπέλαση)
 - Στατικά δεδομένα
 - Διάσχιση πινάκων
 - Σταθερές τιμές
- Υποστήριξη ανάλογα με αρχιτεκτονική συνόλου εντολών

Μέθοδοι προσπέλασης μνήμης (addressing modes)

- Στο σχηματισμό της διεύθυνσης μνήμης μπορούν να συμμετέχουν:
 - Απόλυτες τιμές διεύθυνσης
 - Καταχωρητές
 - Σταθερές τιμές μετατόπισης (offsets)

<i>displacement</i>	mem[offs+reg]	τοπικές
<i>register indirect</i>	mem[reg]	δείκτες
<i>indexed</i>	mem[reg1+reg2]	πίνακες
<i>direct</i>	mem[addr]	στατικές
<i>memory indirect</i>	mem[mem[reg]]	*δείκτες
<i>auto-increment</i>	mem[reg++]	πίνακες
<i>scaled</i>	mem[offs+reg1+reg2*d]	πίνακες

πιθανή
χρήση



Εντολές διακλάδωσης

- Με συνθήκη
 - `bne R1, R2, +8` // branch if not R1==R2
- Χωρίς συνθήκη, σε απόλυτη διεύθυνση
 - `jump 0xFF97DE00`
- Σχετικά ως προς την τρέχουσα θέση
 - `jump +130` // τρέχουσα θέση + offset (+130)
 - Ο παραγόμενος κώδικας μπορεί να τοποθετηθεί οπουδήποτε στη μνήμη



- Εντολές διακλάδωσης είναι και οι κλήσεις συναρτήσεων και οι επιστροφές από αυτές