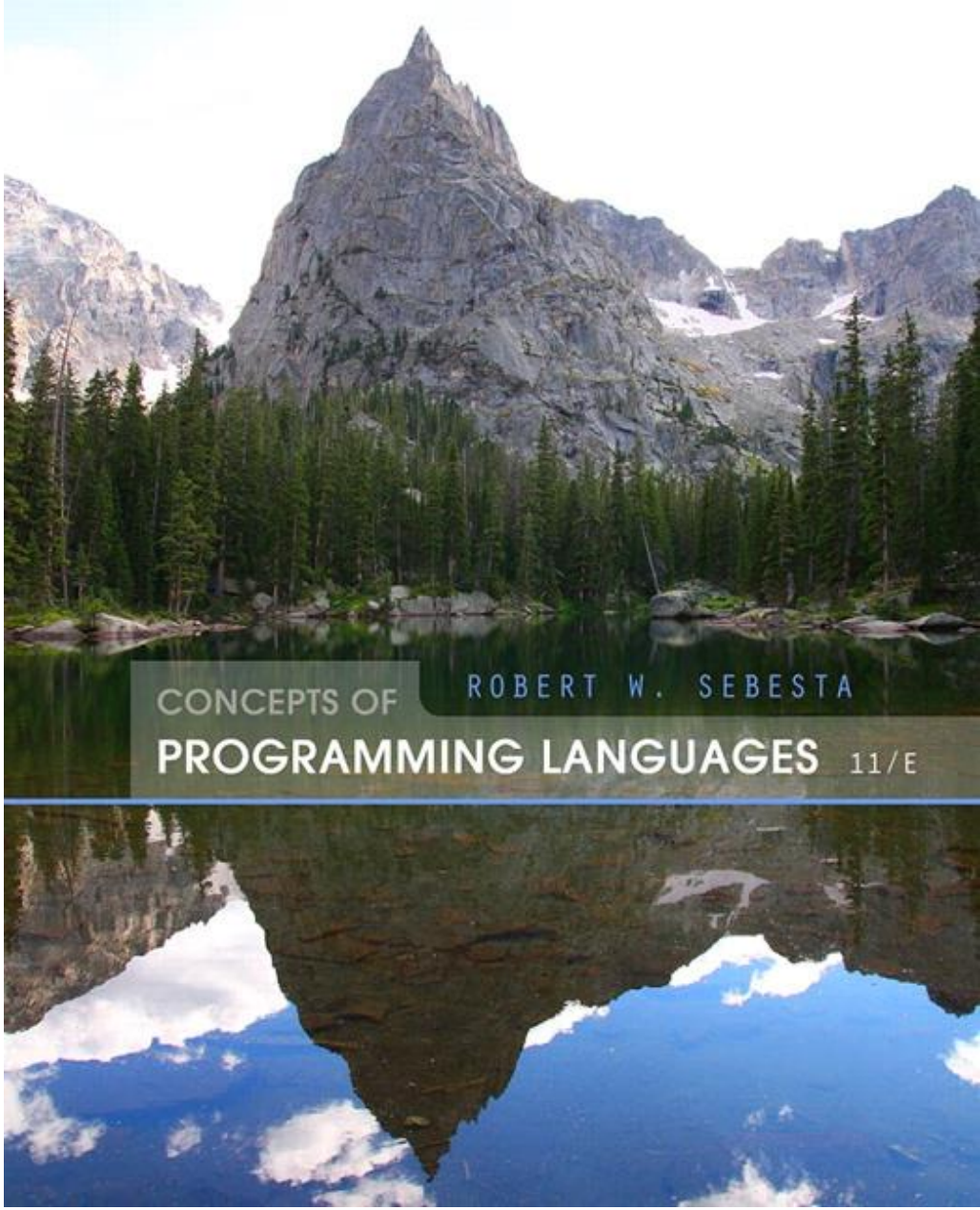
Εισαγωγή

Βασικές έννοιες

Γκόγκος Χρήστος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)
Πανεπιστήμιο Ιωαννίνων



ISBN 0-321-49362-1

Θεματικές Ενότητες – Κεφάλαιο 1

- Για ποιους λόγους να μελετήσει κανείς τις βασικές έννοιες γλωσσών προγραμματισμού;
- Πεδία εφαρμογής γλωσσών προγραμματισμού
- Κριτήρια αποτίμησης γλωσσών
- Παράγοντες που επηρεάζουν τη σχεδίαση των γλωσσών προγραμματισμού
- Κατηγορίες γλωσσών
- Συμβιβασμοί που γίνονται κατά τη σχεδίαση γλωσσών προγραμματισμού
- Μέθοδοι υλοποίησης γλωσσών προγραμματισμού
- Περιβάλλοντα ανάπτυξης προγραμμάτων

Για ποιους λόγους να μελετήσει κανείς τις βασικές έννοιες γλωσσών προγραμματισμού;

- Αυξημένη ικανότητα να εκφράζει ιδέες
- Βελτιωμένο υπόβαθρο για επιλογή, κατά περίπτωση, κατάλληλων γλωσσών
- Αυξημένη ικανότητα να μαθαίνει νέες γλώσσες
- Καλύτερη κατανόηση της σημασίας της υλοποίησης
- Καλύτερη χρήση γλωσσών που ήδη γνωρίζει
- Αναβάθμιση γνώσεων στην επιστήμη υπολογιστών

Πεδία προγραμματισμού

- Επιστημονικές εφαρμογές
  - Μεγάλο πλήθος υπολογισμών με χρήση αριθμών κινητής υποδιαστολής – χρήση διατάξεων (πινάκων)
  - Fortran
- Επιχειρηματικές εφαρμογές
  - Παραγωγή αναφορών, χρήση δεκαδικών αριθμών και χαρακτήρων
  - COBOL
- Τεχνητή Νοημοσύνη (A.I.)
  - Χειρισμός συμβόλων αντί για αριθμούς – χρήση συνδεδεμένων λιστών
  - LISP
- Προγραμματισμός συστημάτων
  - Απαίτηση για υψηλή απόδοση λόγω συνεχούς χρήσης
  - C
- Λογισμικό για το Web
  - Διάφορες επιλογές και συνδυασμοί γλωσσών: markup (π.χ., HTML), σεναρίων (π.χ., PHP), γενικού σκοπού (π.χ., Java)

Κριτήρια αποτίμησης γλωσσών

- **Αναγνωσιμότητα**: η ευκολία με την οποία μπορούν να αναγνωστούν τα προγράμματα και να κατανοηθούν
- **Ευκολία γραφής**: η ευκολία με την οποία η γλώσσα μπορεί να χρησιμοποιηθεί για τη δημιουργία προγραμμάτων
- **Αξιοπιστία**: συμμόρφωση με τις προδιαγραφές (δλδ, κατά πόσο η γλώσσα λειτουργεί όπως περιγράφεται στις προδιαγραφές της)
- **Κόστος**: αφορά το συνολικό κόστος που σχετίζεται με την ανάπτυξη εφαρμογών στη γλώσσα

Κτιτήρια αποτίμησης: Αναγνωσιμότητα

- Συνολική απλότητα
  - Ένα διαχειρίσιμο σύνολο χαρακτηριστικών και προγραμματιστικών κατασκευών
  - Ελάχιστη πολλαπλότητα χαρακτηριστικών
  - Ελάχιστη υπερφόρτωση τελεστών
- Ορθογωνικότητα (Orthogonality)
  - Ένα σχετικά μικρό σύνολο από βασικές προγραμματιστικές κατασκευές μπορούν να συνδυαστούν σε ένα σχετικά μικρό αριθμό τρόπων
  - Οποιοσδήποτε συνδυασμός είναι έγκυρος
- Τύποι δεδομένων
  - Κατά πόσο υπάρχουν επαρκείς προκαθορισμένοι τύποι δεδομένων
- Θέματα σχετικά με τη σύνταξη
  - Μορφές αναγνωριστικών: ευέλικτη σύνταξη
  - Ειδικές λέξεις και μέθοδοι για τη δημιουργία σύνθετων εντολών
  - Μορφή και νόημα: αυτό-περιγραφόμενες προγραμματιστικές κατασκευές, λέξεις κλειδιά με εύκολα κατανοητό νόημα

Κριτήρια αποτίμησης: Απλότητα Γραφής

- Απλότητα και ορθογωνικότητα
  - Λίγες προγραμματιστικές κατασκευές, μικρός αριθμός πρωτογενών στοιχείων, μικρό σύνολο κανόνων για το συνδυασμό τους
- Υποστήριξη αφαίρεσης
  - Η ικανότητα να ορίζονται και να χρησιμοποιούνται σύνθετες δομές ή λειτουργίες με τέτοιο τρόπο που να επιτρέπει να αγνοηθούν οι λεπτομέρειες
- Εκφραστικότητα
  - Ένα σύνολο από σχετικά εύχρηστους τρόπους ορισμού λειτουργιών
  - Ισχύς και πλήθος τελεστών και προκαθορισμένων συναρτήσεων

Κριτήρια αποτίμησης: Αξιοπιστία

- Έλεγχος τύπων (type checking)
  - Έλεγχος για σφάλματα τύπων
- Χειρισμός εξαιρέσεων
  - Παρεμπόδιση σφαλμάτων χρόνου εκτέλεσης και λήψη διορθωτικών μέτρων
- Ψευδώνυμα (aliasing)
  - Παρουσία δύο ή περισσότερων διακριτών τρόπων αναφοράς στην ίδια θέση μνήμης
- Αναγνωσιμότητα και γραφή
  - Μια γλώσσα που δεν υποστηρίζει «φυσικούς» τρόπους έκφρασης ενός αλγορίθμου θα απαιτήσει «μη φυσικές» προσεγγίσεις, και συνεπώς θα μειωθεί η αναγνωσιμότητα

Κριτήρια αποτίμησης: Κόστος

- Εκπαίδευση προγραμματιστών στη χρήση της γλώσσας
- Συγγραφή προγραμμάτων (εγγύτητα προς συγκεκριμένες εφαρμογές)
- Μεταγλώττιση προγραμμάτων
- Εκτέλεση προγραμμάτων
- Σύστημα υλοποίησης της γλώσσας: διαθεσιμότητα ελεύθερων μεταγλωττιστών
- Αξιοπιστία: η χαμηλή αξιοπιστία οδηγεί σε υψηλά κόστη
- Συντήρηση προγραμμάτων

Κριτήρια αποτίμησης: Άλλα

- Φορητότητα
  - Η ευκολία με την οποία προγράμματα μπορούν να μεταφερθούν από μια υλοποίηση σε μια άλλη
- Γενικότητα
  - Εφαρμοσιμότητα σε ευρύ φάσμα εφαρμογών
- Καλά ορισμένη
  - Η πληρότητα και η ακρίβεια του επίσημου ορισμού της γλώσσας

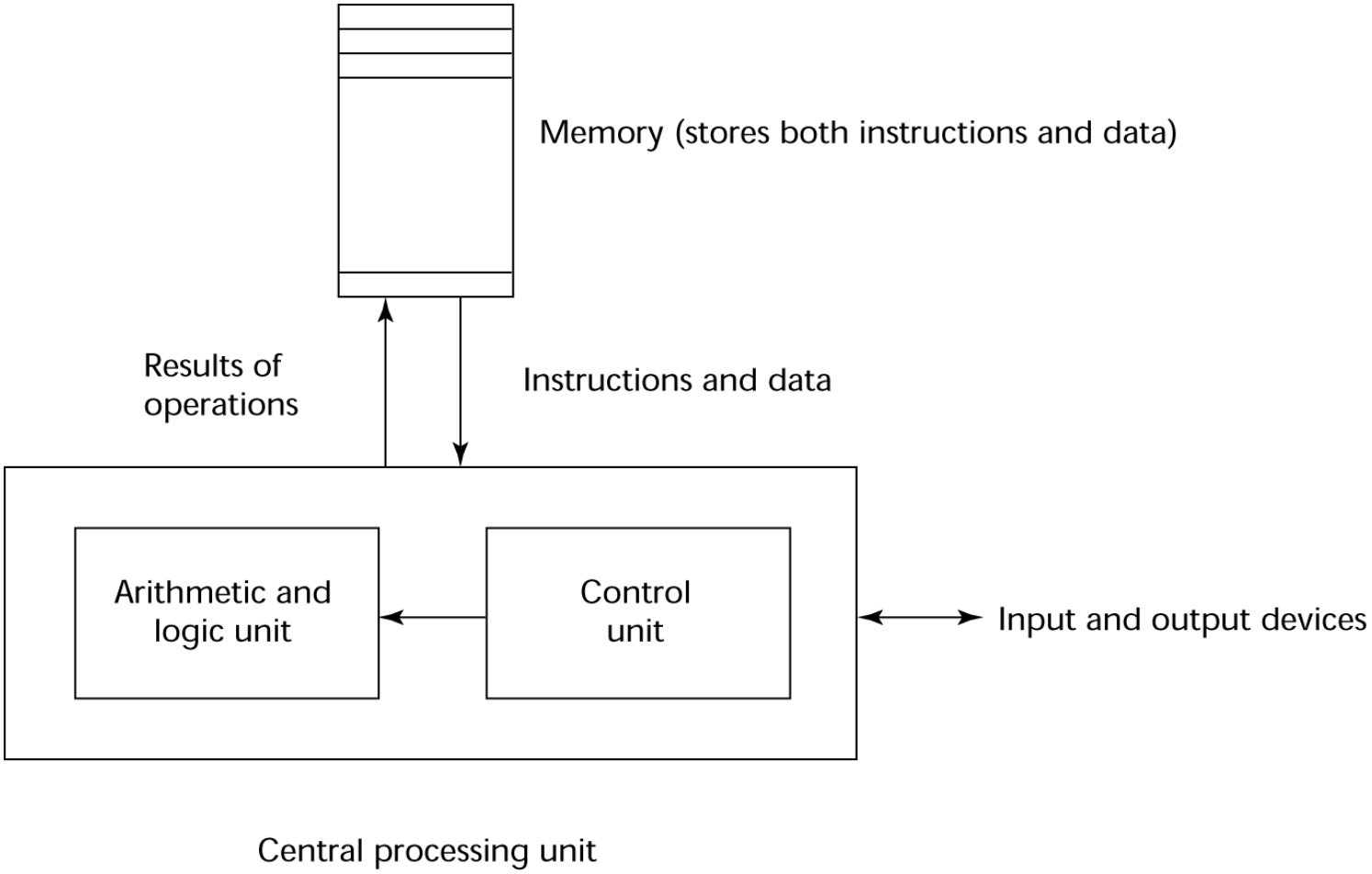
Επιρροές στη Σχεδίαση Γλωσσών

- Αρχιτεκτονική Υπολογιστών
  - Οι γλώσσες αναπτύσσονται με κέντρο την επικρατούσα αρχιτεκτονική υπολογιστών, γνωστή και ως αρχιτεκτονική *von Neumann.*
- Μεθοδολογίες Σχεδίασης Προγραμμάτων
  - Νέες μεθοδολογίες σχεδίασης λογισμικού (π.χ., αντικειμενοστραφής ανάπτυξη λογισμικού) έχουν οδηγήσει σε νέα προγραμματιστικά παραδείγματα και κατ' επέκταση, σε νέες γλώσσες προγραμματισμού

Επιρροή από την Αρχιτεκτονική Υπολογιστών

- Πλέον διαδεδομένη αρχιτεκτονική υπολογιστών: Von Neumann
- Οι προστακτικές γλώσσες, είναι οι πλέον επικρατέστερες λόγω των von Neumann υπολογιστών
  - Τα δεδομένα και τα προγράμματα αποθηκεύονται στη μνήμη
  - Η μνήμη βρίσκεται ξεχωριστά από τη CPU
  - Οι εντολές και τα δεδομένα διοχετεύονται από τη μνήμη στη CPU
  - Βάση προστακτικών γλωσσών
    - Οι μεταβλητές μοντελοποιούν τα κελιά μνήμης
    - Οι εντολές ανάθεσης μοντελοποιούν τη διοχέτευση
    - Οι επαναλήψεις είναι αποδοτικές

Η αρχιτεκτονική von Neumann



Η αρχιτεκτονική von Neumann

- Κύκλος Ανάκλησης-Εκτέλεσης (fetch-execute)

```
αρχικοποίησε το μετρητή προγράμματος (PC)
repeat για πάντα
  ανάκληση της εντολής στην οποία δείχνει ο PC
  μοναδιαία αύξηση του PC
  αποκωδικοποίηση της εντολής
  εκτέλεση της εντολής
end repeat
```

Επιρροή των μεθοδολογιών προγραμματισμού

- 1950s και νωρίς 1960s: Απλές εφαρμογές – κύρια ανησυχία, η αποδοτικότητα της μηχανής
- Τέλος 1960s: Η αποδοτικότητα των ανθρώπων καθίσταται σημαντική – αναζήτηση αναγνωσιμότητας και καλύτερων προγραμματιστικών κατασκευών
  - δομημένος προγραμματισμός
  - Από πάνω προς τα κάτω (top-down) σχεδίαση και βηματική εκλέπτυνση
- Τέλος 1970s: Μετάβαση από προσανατολισμό σε εργασίες σε προσανατολισμό σε δεδομένα
  - αφαίρεση δεδομένων
- Μέσα 1980s: Αντικειμενοστραφής προγραμματισμός
  - Αφαίρεση δεδομένων + κληρονομικότητα + πολυμορφισμός

Κατηγορίες γλωσσών

- Προστακτικές
  - Τα κύρια χαρακτηριστικά είναι οι μεταβλητές, οι εντολές ανάθεσης, και η επανάληψη
  - Περιέχει γλώσσες που υποστηρίζουν τον αντικειμενοστραφή προγραμματισμό
  - Περιέχει γλώσσες σεναρίων
  - Περιέχει οπτικές γλώσσες
  - Παραδείγματα: C, Java, Perl, JavaScript, Visual BASIC .NET, C++
- Συναρτησιακές
  - Ο κύριος τρόπος με τον οποίο γίνονται υπολογισμοί είναι εφαρμόζοντας συναρτήσεις σε παραμέτρους που δίνονται
  - Παραδείγματα: LISP, Scheme, ML, F#
- Λογικές
  - Βασίζονται σε κανόνες (οι κανόνες ορίζονται χωρίς καμία συγκεκριμένη σειρά)
  - Παράδειγμα: Prolog
- Υβριδικές γλώσσες markup (γλώσσες σήμανσης)/προγραμματισμού
  - Markup επεκτάσεις γλωσσών που υποστηρίζουν σε κάποιο βαθμό προγραμματισμό
  - Παραδείγματα: JSTL, XSLT

Συμβιβασμοί κατά τη Σχεδίαση Γλωσσών

- Αξιοπιστία vs. Κόστος εκτέλεσης
  - Παράδειγμα: Η Java απαιτεί όλες οι αναφορές σε στοιχεία διατάξεων να ελέγχονται ότι πραγματοποιούν έγκυρη δεικτοδότηση, το οποίο οδηγεί σε αυξημένο κόστος εκτέλεσης
- Ευκολία Ανάγνωσης vs. Ευκολία Γραφής

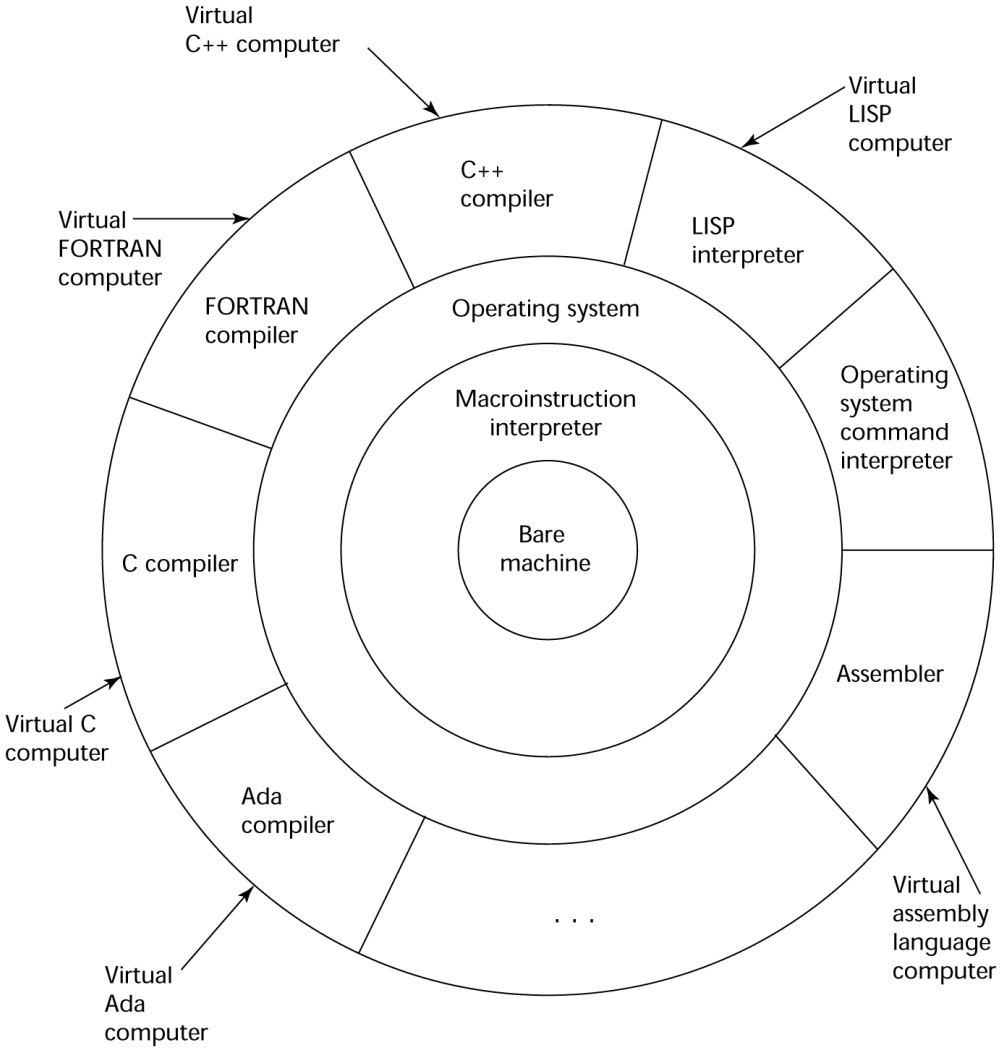
  Παράδειγμα: Η γλώσσα APL παρέχει πολλούς ισχυρούς τελεστές (και ένα μεγάλο αριθμό νέων συμβόλων), που επιτρέπουν σύνθετους υπολογισμούς να γραφούν σε συνοπτικά προγράμματα, αλλά χάνει σε ευκολία γραφής
- Γραφή (ευελιξία) vs. Αξιοπιστία
  - Παράδειγμα: Οι δείκτες C++ είναι ισχυροί και πολύ ευέλικτοι, αλλά τα προγράμματα που τους χρησιμοποιούν μπορεί να μην είναι αξιόπιστα

Μέθοδοι υλοποίησης

- Μεταγλώττιση
  - Τα προγράμματα μεταφράζονται σε γλώσσα μηχανής – συμπεριλαμβάνει και τα JIT συστήματα
  - Χρήση: Μεγάλες εμπορικές εφαρμογές
- Καθαρή διερμηνεία
  - Τα προγράμματα διερμηνεύονται από ένα άλλο πρόγραμμα, τον διερμηνευτή
  - Χρήση: Μικρά προγράμματα ή όταν η αποδοτικότητα είναι δευτερεύουσας σημασίας
- Υβριδικά συστήματα υλοποίησης
  - Πρόκειται για συμβιβασμό ανάμεσα σε μεταγλωττιστές και σε καθαρούς διερμηνευτές
  - Χρήση: Μικρά και μεσαίου μεγέθους συστήματα ή όταν η αποδοτικότητα είναι δευτερεύουσας σημασίας

Οργάνωση σε Επίπεδα

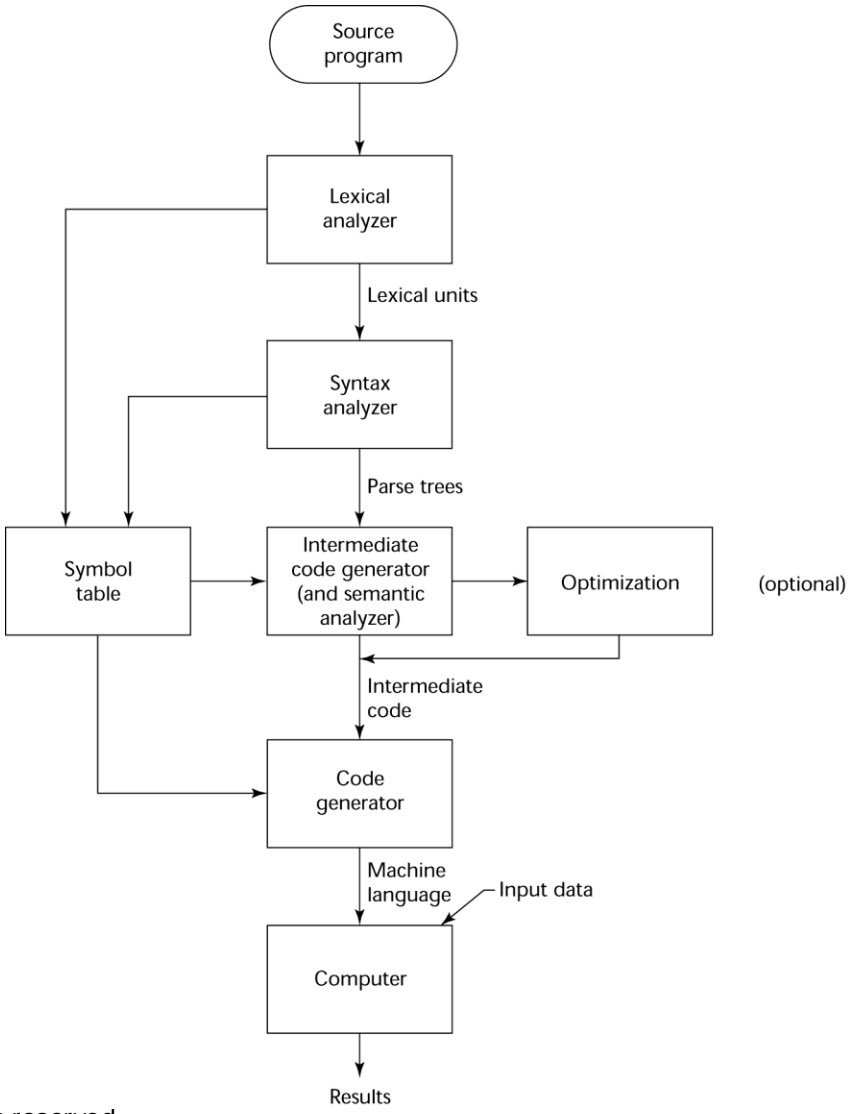
Το λειτουργικό σύστημα και η υλοποίηση της γλώσσας βρίσκονται σε επίπεδα πάνω από το υλικό του υπολογιστή



Μεταγλώττιση

- Μεταφράζει προγράμματα υψηλού-επιπέδου (πηγαία γλώσσα) σε κώδικα μηχανής (γλώσσα μηχανής)
- Χρονοβόρα μετάφραση, γρήγορη εκτέλεση
- Η διαδικασία της μεταγλώττισης έχει πολλές φάσεις:
  - Λεκτική ανάλυση: μετασχηματίζει τους χαρακτήρες του πηγαίου προγράμματος σε λεκτικές μονάδες
  - Συντακτική ανάλυση: μετασχηματίζει τις λεκτικές μονάδες σε δένδρα συντακτικής ανάλυσης (parse trees) που αναπαριστούν τη συντακτική δομή του προγράμματος
  - Σημασιολογική ανάλυση: δημιουργία ενδιάμεσου κώδικα
  - Γεννήτρια κώδικα: παράγει τον κώδικα μηχανής

Η διαδικασία μεταγλώττισης



Επιπλέον ορολογία μεταγλώττισης

- **Φόρτωση module** (executable image): ο κώδικας χρήστη και ο κώδικας συστήματος μαζί
- **Σύνδεση και φόρτωση**: η διαδικασία συλλογής προγραμματιστικών μονάδων συστήματος και η σύνδεσή τους σε ένα πρόγραμμα χρήστη

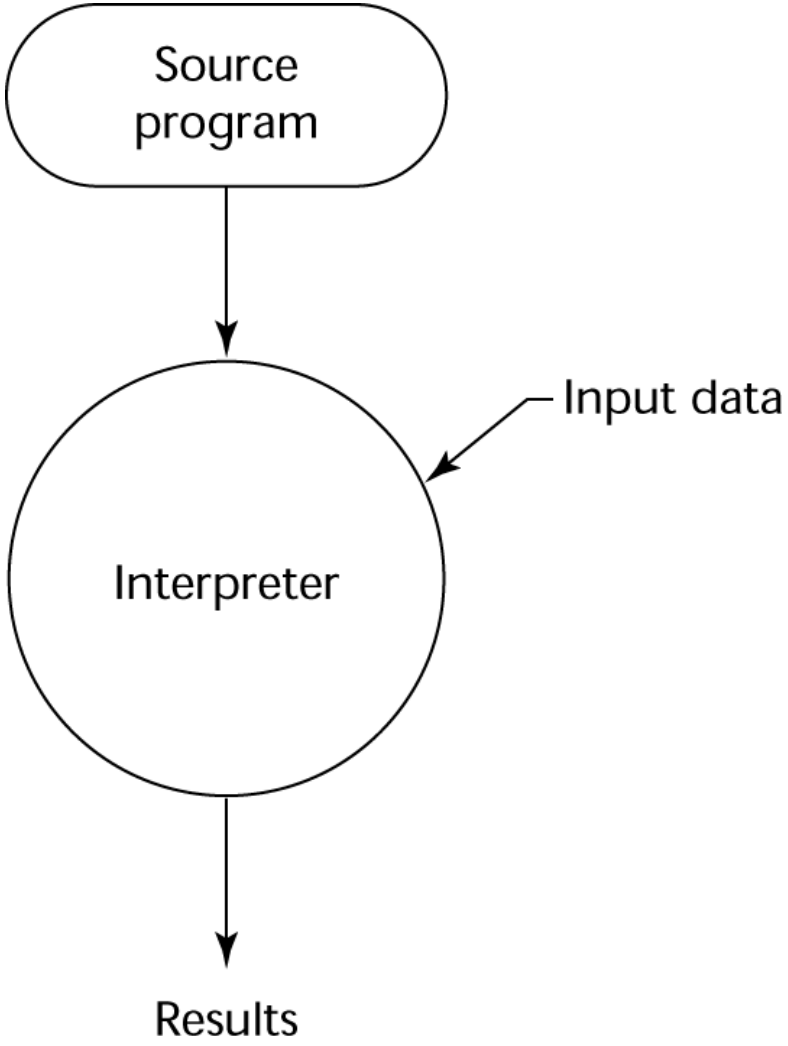
Η συμφόρηση Von Neumann

- Η ταχύτητα σύνδεσης ανάμεσα στη μνήμη του υπολογιστή και τον επεξεργαστή του καθορίζει την ταχύτητα του υπολογιστή
- Οι εντολές του προγράμματος μπορούν συχνά να εκτελεστούν πολύ ταχύτερα από την ταχύτητα της σύνδεσης – συνεπώς η ταχύτητα της σύνδεσης μνήμη–επεξεργαστή αποτελεί το σημείο συμφόρησης (*bottleneck)*
- Είναι γνωστό ως *von Neumann bottleneck* – αποτελεί τον κύριο περιοριστικό παράγοντα για την ταχύτητα των υπολογιστών

Καθαρή Διερμηνεία

- Δεν γίνεται μετάφραση
- Ευκολότερη υλοποίηση προγραμμάτων (τα σφάλματα χρόνου εκτέλεσης μπορούν εύκολα και άμεσα να εντοπιστούν)
- Βραδύτερη εκτέλεση (10 έως100 φορές βραδύτερα από τα μεταγλωττισμένα προγράμματα)
- Συχνά απαιτεί περισσότερο αποθηκευτικό χώρο
- Πλέον συναντάται σπάνια για τις παραδοσιακές γλώσσες υψηλού επιπέδου
- Σημαντική επιστροφή της καθαρής διερμηνείας σε ορισμένες Web scripting γλώσσες (π.χ., JavaScript, PHP)

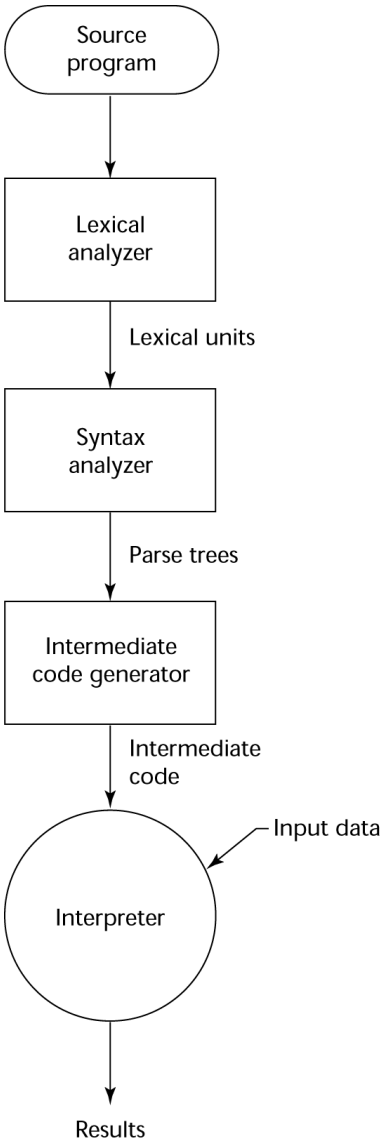
Διαδικασία καθαρής διερμηνείας



Συστήματα Υβριδικής Υλοποίησης

- Συμβιβασμός ανάμεσα σε μεταγλωττιστές και καθαρούς διερμηνευτές
- Ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου μεταφράζεται σε ενδιάμεση γλώσσα που επιτρέπει τη γρήγορη διερμηνεία
- Ταχύτερο από την καθαρή διερμηνεία
- Παραδείγματα
  - Τα προγράμματα σε Perl μεταγλωττίζονται μερικώς για τον εντοπισμό λαθών πριν τη διερμηνεία
  - Οι αρχικές υλοποιήσεις της Java ήταν υβριδικές – η ενδιάμεση μορφή (byte code) παρέχει φορητότητα σε οποιαδήποτε μηχανή διαθέτει διερμηνευτή byte code και ένα σύστημα χρόνου εκτέλεσης (και τα δύο μαζί ονομάζονται *Java Virtual Machine*)

Διαδικασία Υβριδικής Υλοποίησης



Συστήματα Υλοποίησης Just-in-Time

- Αρχικά μεταφράζουν τα προγράμματα σε μια ενδιάμεση γλώσσα
- Στη συνέχεια, η ενδιάμεση γλώσσα στην οποία βρίσκονται τα υποπρογράμματα μεταγλωττίζεται σε γλώσσα μηχανής όταν αυτά καλούνται
- Η έκδοση του μεταγλωττισμένου κώδικα μηχανής διατηρείται για μεταγενέστερες κλήσεις για λόγους ταχύτητας εκτέλεσης
- Τα JIT συστήματα χρησιμοποιούνται ευρέως στα προγράμματα Java
- Οι γλώσσες .NET χρησιμοποιούν ένα JIT σύστημα
- Στην ουσία, τα JIT συστήματα είναι μεταγλωττιστές με καθυστέρηση

Προεπεξεργαστές

- Οι μακροεντολές προεπεξεργαστή συχνά χρησιμοποιούνται για να καθορίσουν κώδικα από άλλο αρχείο που πρόκειται να συμπεριληφθεί στον πηγαίο κώδικα
- Ένας προεπεξεργαστής επεξεργάζεται ένα πρόγραμμα αμέσως πριν μεταγλωττιστεί και επεκτείνει τις μακροεντολές προεπεξεργαστή
- Γνωστό παράδειγμα προεπεξεργαστή: Ο προεπεξεργαστής της C
  - επεκτείνει τις μακροεντολές `#include`, `#define`, και άλλες

Προγραμματιστικά περιβάλλοντα

- Μια συλλογή εργαλείων που χρησιμοποιούνται για ανάπτυξη λογισμικού
- UNIX
  - Παλαιότερο λειτουργικό σύστημα και συλλογή εργαλείων
  - Σήμερα χρησιμοποιούνται συχνά μέσω γραφικών διεπαφών GUI (π.χ., CDE, KDE, or GNOME) που εκτελείται πάνω από το UNIX
- Microsoft Visual Studio.NET
  - Ένα μεγάλο, σύνθετο οπτικό (visual) περιβάλλον ανάπτυξης εφαρμογών
  - Χρησιμοποιείται για τη δημιουργία Web και non-Web εφαρμογών σε οποιαδήποτε .NET γλώσσα
- NetBeans
  - Παρόμοιο με το Visual Studio .NET, αλλά για εφαρμογές σε Java

Σύνοψη

- Η μελέτη των γλωσσών προγραμματισμού είναι σημαντική για πολλούς λόγους:
  - Ενίσχυση χρήσης διαφορετικών προγραμματιστικών κατασκευών
  - Επιτρέπει την έξυπνη επιλογή της κατάλληλης γλώσσας προγραμματισμού κατά περίπτωση
  - Καθιστά την εκμάθηση νέων γλωσσών ευκολότερη
- Τα πλέον σημαντικά κριτήρια αξιολόγησης γλωσσών προγραμματισμού είναι:
  - Αναγνωσιμότητα, ευκολία γραφής, αξιοπιστία, κόστος
- Οι κύριες επιρροές που δέχθηκε η σχεδίαση γλωσσών ήταν από την αρχιτεκτονική υπολογιστών και από τις μεθοδολογίες ανάπτυξης λογισμικού
- Οι κύριες μέθοδοι υλοποίησης γλωσσών προγραμματισμού είναι: μεταγλώττιση, καθαρή διερμηνεία, και υβριδική υλοποίηση