

Λογισμικό Συστήματος

Κλειώ Σγουροπούλου

Λογισμικό συστήματος

- Λειτουργικό σύστημα
- Μεταφραστές γλώσσας (translators)
 - Διερμηνείς (interpreters)
 - Μεταγλωττιστές (compilers)
- Εκδότες (editors)
- Φορτωτές (loaders)
- Συνδέτες (linkers)
- Λογισμικό επικοινωνίας

ΓΠΥΕ vs. ΓΜ

- Γλώσσες προγραμματισμού υψηλού επιπέδου
 - Ευκολότερη κατανόηση
 - Προστασία από λάθη κατά την έκφραση αλγορίθμων
 - Μεταφερισιμότητα / φορητότητα
- Γλώσσες Μηχανής
 - Γρηγορότερα και συντομότερα προγράμματα
 - Άμεση πρόσβαση σε στοιχεία του υλικού
 - Συντονισμός

Διερμηνέας

- Ο διερμηνέας είναι ένα πρόγραμμα το οποίο διαβάζει, μεταφράζει και εκτελεί **δήλωση προς δήλωση** προγράμματα που έχουν γραφτεί σε μια γλώσσα υψηλού επιπέδου

αρχή προγράμματος υψηλού επιπέδου

repeat

μετάφρασε την επόμενη δήλωση υψηλού επιπέδου

if κανένα συντακτικό λάθος

then εκτέλεσε

else ανάφερε λάθος

until τέλος προγράμματος υψηλού επιπέδου ή συντακτικό λάθος

Μεταγλωττιστής

- Ο μεταγλωττιστής είναι ένα πρόγραμμα το οποίο διαβάζει προγράμματα που έχουν γραφτεί σε μια γλώσσα υψηλού επιπέδου– την **πηγαία** (*source*) γλώσσα – και τα μεταφράζει σε ισοδύναμα προγράμματα σε μια άλλη γλώσσα – γλώσσα **μεταφοράς** (*target*)

αρχή προγράμματος υψηλού επιπέδου

repeat

μετάφρασε την επόμενη δήλωση υψηλού επιπέδου

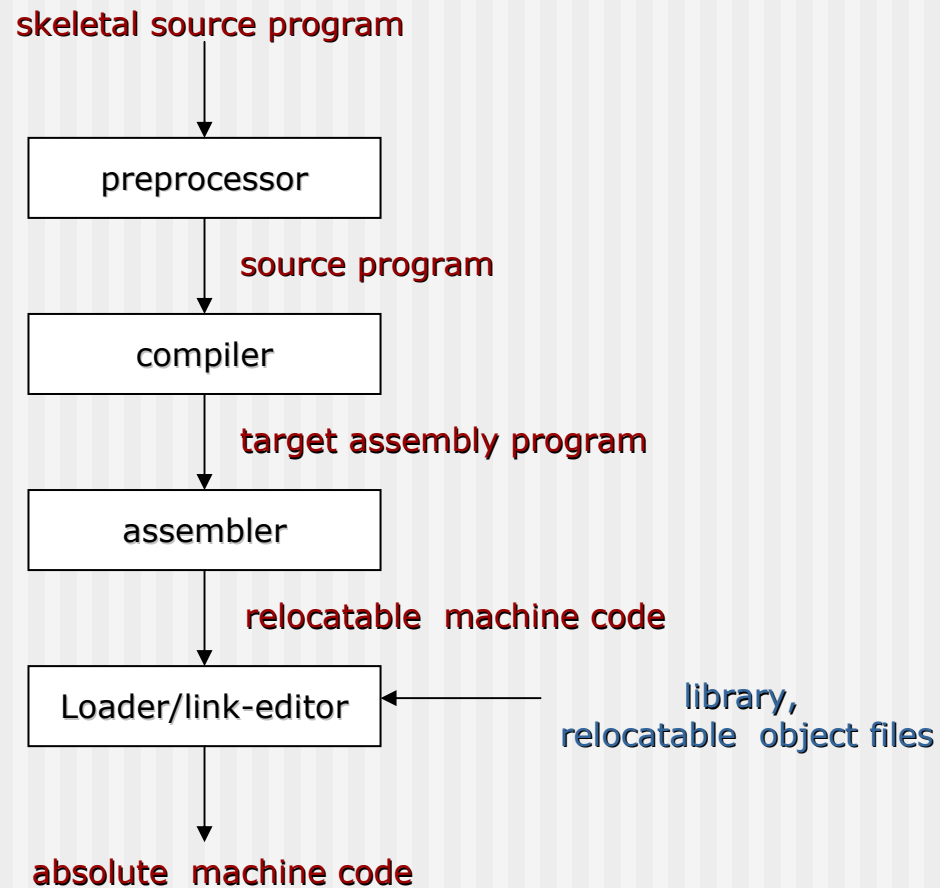
until *τέλος προγράμματος υψηλού επιπέδου*

εκτέλεση ολόκληρου του μεταφρασμένου προγράμματος

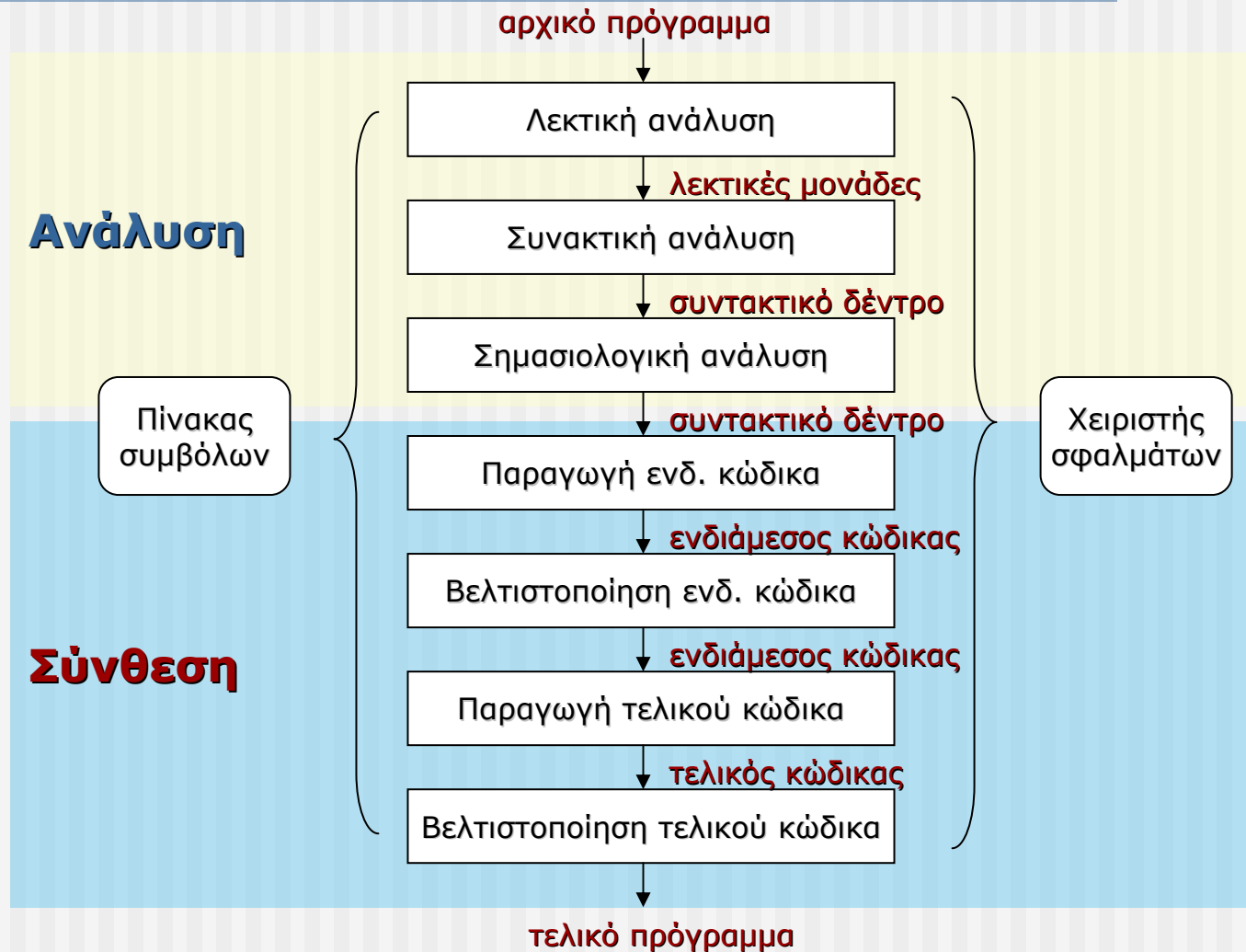
Μεταφραστές vs. Μεταγλωττιστές

- **Μεταφραστές**
 - Πολλαπλή μετάφραση ίδιων δηλώσεων προγράμματος (επιβάρυνση στην ταχύτητα εκτέλεσης)
 - Μεγαλύτερη ικανότητα εντοπισμού σφαλμάτων
 - Ενδείκνυται κατά την ανάπτυξη
- **Μεταγλωττιστές**
 - Επαναληπτική εκτέλεση μεταγλωττισμένων προγραμμάτων
 - Ταχύτητα
 - Ενδείκνυται για παραγωγική χρήση

Το “περιβάλλον” του μεταγλωττιστή



Φάσεις και προϊόντα μεταγλώττισης



Παράδειγμα

■ Αρχείο εισόδου

"test.c"

■ Αρχείο εξόδου

"test.out"

```
int A;  
int B;  
main()  
{  
    A = B+123;  
}
```

```
.comm A,4  
.comm B,4
```

main:

```
mov B,d0  
mov #123,d1  
add d1,d0  
mov d0,A  
ret
```

Λεξική ανάλυση (scanning)

- Κατασκευή **λεκτικών μονάδων** (lexemes) από χαρακτήρες εισόδου πηγαίου προγράμματος (Regular expressions, transition diagrams)
- Κατηγοριοποίηση λεκτικών μονάδων σε **κουπόνια** (tokens)
- Συστατικά κουπονιών:
 - **Τύπος**: Τα κουπόνια αποτελούν ακολουθίες χαρακτήρων που εκφράζουν μια γενική έννοια: λέξη κλειδί (**keyword**), όνομα (**identifier**), σταθερά (**constant**), τελεστής (**operator**) και διαχωριστής (**delimiter**)
 - **Τιμή** π.χ. (**CONST**, '123') (**IDENT**, 'A')

Λεξική ανάλυση (scanning)

- Αφαίρεση κενών και περιττών χαρακτήρων (tabs, new lines)
- Αφαίρεση των σχολίων
- Έλεγχος λαθών
 - Έλεγχος μεγέθους identifiers/string constants, βάσει κανόνων γλώσσας
 - Έλεγχος μη νόμιμων identifiers
 - Έλεγχος λανθασμένων αναθέσεων

Παράδειγμα: Λεξική ανάλυση

- *Είσοδος: ακολουθία χαρακτήρων*

```
intsp A; nl intsp B; nl main() nl { nl Asp = Bsp + 123; nl } nl
```

- *Έξοδος: ακολουθία κουπονιών*

```
INTtok IDENTtok ()
SEMICOLONtok INTtok IDENTtok ()
SEMICOLONtok IDENTtok ()
LBRACKETtok RBRACKETtok
LCURLYtok IDENTtok ()
Eqtok IDENTtok ()
PLUSTok CONSTtok 123
SEMICOLONtok RCURLYtok
```

Symbol Table	
A	
B	
main	

Γραμματικές

- Ο ορισμός μιας γλώσσας προγραμματισμού περιλαμβάνει:
 - **Συντακτικό:** κανόνες για την κατασκευή ‘αποδεκτών’ διατυπώσεων. Ο τρόπος εφαρμογής των συντακτικών κανόνων καθορίζει τη συντακτική ορθότητα ενός προγράμματος
 - **Σημασιολογία:** μοντελοποίηση δεδομένων και λειτουργιών, ισοδυναμίες προγραμμάτων, κ.λπ.

Γραμματικές

- **Συντακτικό**: συνήθως καθορίζεται από *γραμματικές χωρίς συμφραζόμενα* (ΓΧΣ) (context-free grammars / BNF)
- Παράδειγμα BNF **παραγωγής**:
`stat -> if (exp) stat else stat`
τερματικά: `if, (,), else` μη-τερματικά: `stat, exp`
- Κάθε **παραγωγή** αποτυπώνει έναν έγκυρο τρόπο με βάση τον οποίο τα μη-τερματικά μπορούν να αναπτυχθούν σε συμβολοσειρές τερματικών και μη-τερματικών

Ορισμός ΓΧΣ

- Μια ΓΧΣ περιλαμβάνει:
 - t : σύνολο κουπονιών (‘τερματικά’)
 - nt : σύνολο μη-τερματικών
 - P : σύνολο παραγωγών
 - S : ένα μη-τερματικό αρχικό σύμβολο
- Παράδειγμα

bin \rightarrow **bin** + **dig**

bin \rightarrow **bin** - **dig**

bin \rightarrow **dig**

dig \rightarrow **0**

dig \rightarrow **1**

bin \rightarrow **bin** + **dig** |
bin - **dig** | **dig**

dig \rightarrow **0** | **1**

Ορισμός ΓΧΣ

- Με αφετηρία το αρχικό σύμβολο και συνεχή αντικατάσταση κάθε μη-τερματικού με το δεξί μέρος της αντίστοιχης παραγωγής, προκύπτουν συμβολοσειρές τερματικών
- Μια τέτοια συμβολοσειρά μη-τερματικών καλείται **προτασιακή μορφή**
- Η **γλώσσα** μιας γραμματικής είναι το σύνολο όλων των προτασιακών μορφών που μπορούν να παραχθούν από το αρχικό της σύμβολο

Συντακτική ανάλυση (parsing)

- Έλεγχος γραμματικής ορθότητας προγράμματος εισόδου
- Κατασκευή αντίστοιχου **συντακτικού δένδρου** (Abstract Syntax Tree)
- Δυο γενικές κατηγορίες συντακτικών αναλυτών
 - "συντακτικοί αναλυτές από πάνω προς τα κάτω" (top-down parsers)
 - "συντακτικοί αναλυτές από κάτω προς τα πάνω" (bottom-up parsers)

Συντακτική ανάλυση: top-down

- Εκκίνηση από ρίζα δένδρου (αρχικό σύμβολο γραμματικής)
- πορεία προς τα φύλλα (συμβολοσειρά εισόδου)
- με διαδοχικές αντικαταστάσεις των μη-τερματικών συμβόλων
- χρησιμοποιώντας τους συντακτικούς κανόνες της γραμματικής από αριστερά προς τα δεξιά

Συντακτική ανάλυση: top-down

1. Ξεκίνα από το ΑΣ
2. Εξέτασε κάθε εναλλακτική παραγωγή για το ΑΣ
3. Σύγκρινε το κουπόνι εισόδου με το πρώτο σύμβολο του δεξιού μέρους κάθε εναλλακτικής παραγωγής
4. Αν βρεθεί αντίστοιχη παραγωγή χρησιμοποίησέ τη για επέκταση δένδρου
5. Επανάλαβε για επόμενο κουπόνι εισόδου με στοχο τον καθορισμό παραγωγής για το επόμενο μη τερματικό

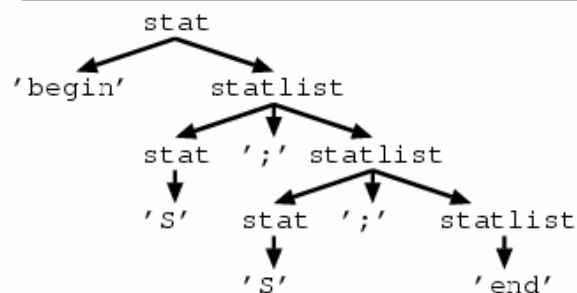
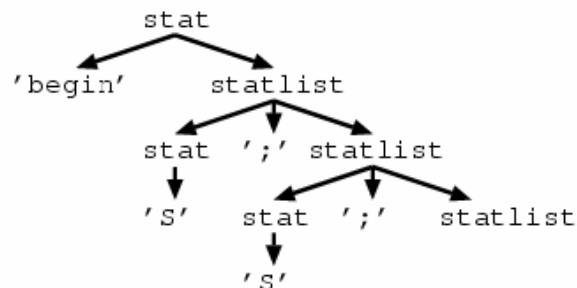
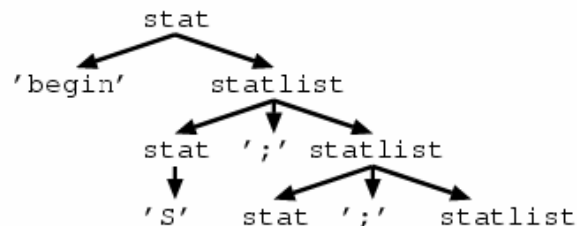
Συντακτική ανάλυση: top-down

Παράδειγμα

stat -> **begin** statlist | **S**
statlist -> **stat ; statlist** | **end**

stat	begin S; S; end
<pre> graph TD stat --> begin["'begin'"] stat --> statlist </pre>	<u>begin</u> S; S; end
<pre> graph TD stat --> begin["'begin'"] stat --> statlist statlist --> stat2["stat ';' statlist"] </pre>	no match; persevere
<pre> graph TD stat --> begin["'begin'"] stat --> statlist statlist --> stat2["stat ';' statlist"] stat2 --> S["'S'"] </pre>	<u>begin</u> S; S; end

Συντακτική ανάλυση: top-down



The final parse tree

All input consumed

Συντακτική ανάλυση: bottom-up

- Εκκίνηση από τα φύλλα (συμβολοσειρά εισόδου)
- πορεία προς τη ρίζα (αρχικό σύμβολο)
- με διαδοχική αναγνώριση δεξιού μέλους συντακτικών κανόνων
- και αντικατάσταση με τα αριστερά τους μέλη.

Συντακτική ανάλυση: bottom-up

- Αναλυτές «**ολίσθησε-ελάττωσε**»
 1. Τοποθέτησε στη στοίβα (**ολίσθησε - shift**) σύμβολα εισόδου μέχρι να εμφανισθεί το δεξιό μέλος κάποιου συντακτικού κανόνα στην κορυφή της
 2. Αφαίρεσε από τη στοίβα (**ελάττωσε - reduce**) αυτό το δεξιό μέλος και στη θέση του τοποθέτησε το αντίστοιχο αριστερό του μέλος.
 3. Επανέλαβε έως ότου σαρωθεί όλη η συμβολοσειρά εισόδου και μείνει στη στοίβα μόνο το αρχικό σύμβολο της γραμματικής.

Συντακτική ανάλυση: bottom-up

Παράδειγμα

Rule A:

stat ->

begin

statlist |

S

Rule B:

statlist ->

stat ;

statlist |

end

<i>Stack</i>	<i>current symbol</i>	<i>Remaining input</i>	<i>Action</i>
	begin	S ; S ; end	shift
begin	<u>S</u>	; S ; end	reduce, rule A
begin	stat	; S ; end	shift
begin stat	;	S ; end	shift
begin stat ;	<u>S</u>	; end	reduce A
begin stat ;	stat	; end	shift
begin stat ; stat	;	end	shift
begin stat ; stat ;	<u>end</u>		reduce B
begin stat ; stat ;	<u>statlist</u>		reduce B
begin stat ;	<u>statlist</u>		reduce B
begin	<u>statlist</u>		reduce A
	stat		finished!

Γέννηση κώδικα

- Εκχώριση ελεύθερης μνήμης στα δεδομένα που διαχειρίζεται το πρόγραμμα
 - Χρήση δηλωτικής πληροφορίας
- Δημιουργία εντολών σε γλώσσα μηχανής
 - για κάθε συντακτικό στοιχείο