Μεταγλωττιστές II

Η οργάνωση του μεταγλωττιστή

Νικόλαος Καββαδίας
nkavv@uop.gr

03 Νοεμβρίου 2010

Αντικείμενο του μαθήματος CST325: Μεταγλωττιστές II (1)

- Επιμέρους στόχοι του μαθήματος
  - Παρουσίαση θεμάτων που άπτονται του σχεδιασμού μεταγλωττιστών πέρα από τα βασικά
    - Τεκμηριωμένες τεχνικές για όλα τα στάδια της μεταγλώττισης
    - Αναλύσεις, μετασχηματισμοί και βελτιστοποιήσεις
    - Τεχνικές εκμετάλλευσης της παραλληλίας σε όλα τα επίπεδα
    - Πρακτικές περιπτώσεις εφαρμογής των τεχνικών αυτών σε πραγματικούς μεταγλωττιστές
  - Άντληση γνώσης από την 'πηγή' (journal, conference and workshop papers and presentations) με υλικό για μελέτη από reading list Αγγλικής βιβλιογραφίας
  - Παρουσίαση της 'εργαλειοθήκης' του σχεδιαστή μεταγλωττιστών

Αντικείμενο του μαθήματος CST325: Μεταγλωττιστές II (2)

- Επιθυμητές (στην πράξη, προαπαιτούμενες) γνώσεις για επιτυχή παρακολούθηση του μαθήματος
  - ☑ Απευθύνεται σε φοιτητές που έχουν διδαχθεί και εξεταστεί επιτυχώς στο μάθημα 'Μεταγλωττιστές' και τα μαθήματα προγραμματισμού 1ου και 2ου εξαμήνου
  - ☑ Η καλή γνώση προγραμματισμού σε κάποια διαδικαστική γλώσσα (C, C++, Java) και δομών δεδομένων, καθώς και γνώσεις αρχιτεκτονικής υπολογιστών είναι χρήσιμες
  - ☑ Εξοικείωση με λογισμικά εργαλεία ανάπτυξης εφαρμογών και ιδιαίτερα εργαλεία GNU όπως GCC, bison, flex είναι θεμιτή αλλά όχι απαραίτητη
  - ☑ Το κυριότερο προσόν είναι η καλή διάθεση και το προσωπικό ενδιαφέρον για το αντικείμενο του μαθήματος

Περίγραμμα μαθήματος

- Η οργάνωση του μεταγλωττιστή
- Γέννηση ενδιάμεσης αναπαράστασης
- Επιλογή κώδικα
- Κατανομή καταχωρητών
- Βελτιστοποιήσεις ανεξάρτητες από την αρχιτεκτονική
- Χρονοπρογραμματισμός κώδικα και παραλληλία επιπέδου εντολών
- Βελτιστοποιήσεις για την ανάδειξη της παραλληλίας εντολών και της τοπικότητας δεδομένων (2 διαλέξεις)
- Περιβάλλον χρόνου εκτέλεσης και γέννηση τελικού κώδικα για αρχιτεκτονικές μηχανής
- Ειδικά θέματα και ασκήσεις
- Επαναστοχεύσιμοι μεταγλωττιστές και εργαλεία ανάπτυξης μεταγλωττιστών

Εισαγωγικά

- Το αντικείμενο των 'Μεταγλωττιστών II' δεν είχε διδαχθεί, με αυτήν τη μορφή, μέχρι τώρα στα ελληνικά πανεπιστήμια
- Η σειρά διαλέξεων που θα πραγματοποιηθεί αντιστοιχεί σε ένα course **Advanced Compiler Techniques** ή **Advanced Compiler Design/Construction** των τμημάτων Computer Science των πανεπιστημίων του εξωτερικού
- Ενημέρωση για ανακοινώσεις, διαλέξεις, ύλη, εργασίες από τον ιστότοπο του μαθήματος:
  `http://eclass.uop.gr/courses/CST325/`

Περιγραφή της γραμματικής μιας πηγαίας γλώσσας

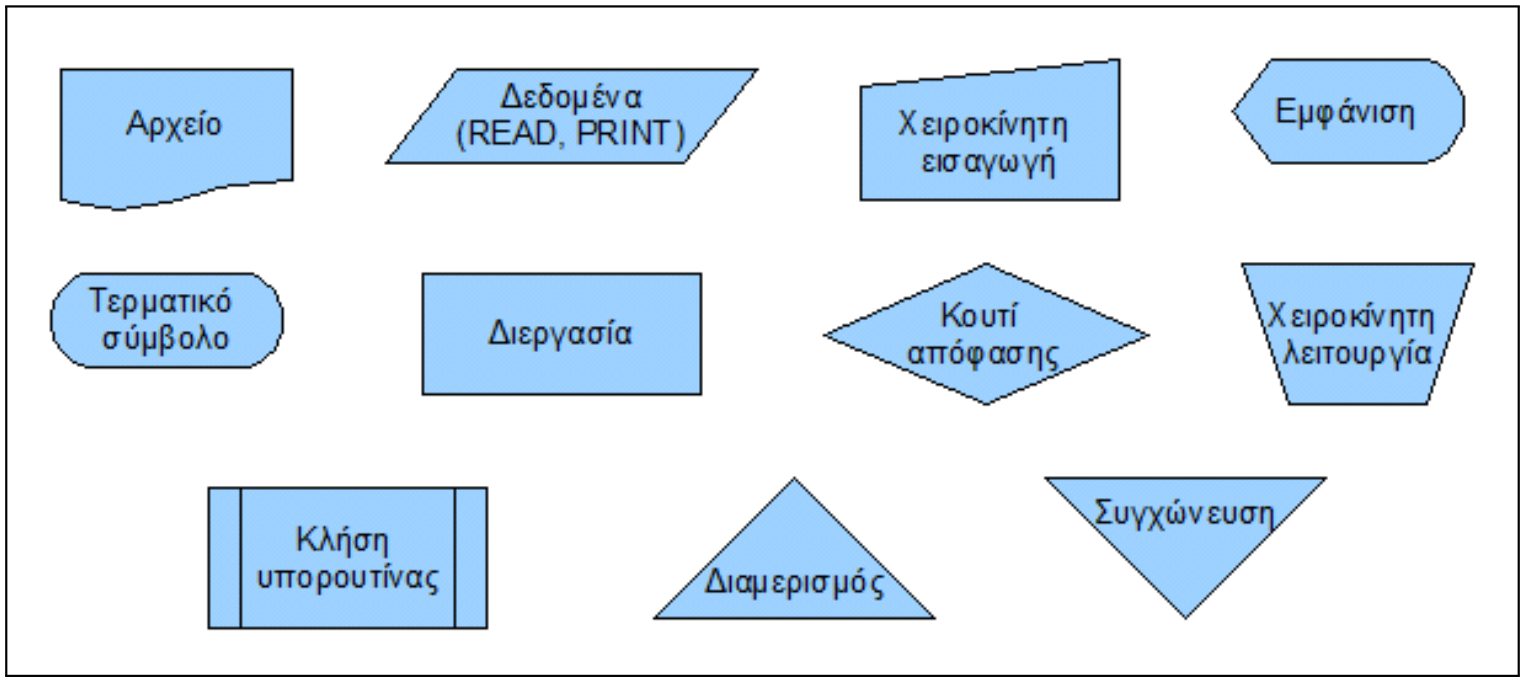
- Η περιγραφή της οργάνωσης μιας πηγαίας γλώσσας γενικά γίνεται με τη διατύπωση της *γραμματικής* της
- Η γραμματική απαρτίζεται από λεκτικούς και συντακτικούς κανόνες
- Σύνθετες δηλώσεις στην πηγαία γλώσσα περιγράφονται με αναδρομική εφαρμογή απλούστερων κανόνων
- Παράδειγμα σε BNF: Backus-Naur Form

```
expr ::= literal
|| expr <+> expr
|| expr <*> expr
```

- Αποδόμηση της εισόδου σε τερματικές (+,∗, literal) και μη-τερματικές (expr) λεκτικές μονάδες
- Συντακτική (και σημασιολογική) ανάλυση με αναγνώριση των γραμματικών κανόνων

Σχεδιασμός διαγράμματος ροής (flow chart)

- Το διάγραμμα ροής αποτελεί ένα είδος αναπαράστασης προγραμματικών δομών σε μια οποιαδήποτε γλώσσα (υψηλού ή χαμηλού επιπέδου)



Η γλώσσα ANSI C [K&R, 1988]

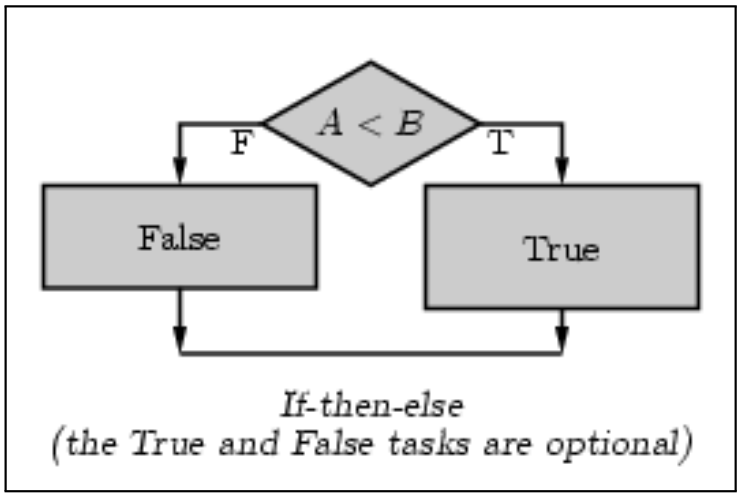
- Η C είναι η συχνότερα χρησιμοποιούμενη γλώσσα για τον προγραμματισμό ενσωματωμένων συστημάτων (embedded systems) και εξακολουθεί να απολαμβάνει εκτίμησης για τον προγραμματισμό συστημάτων (UNIX, GNU software, Linux kernel)
- Ακολουθεί το διαδικαστικό μοντέλο προγραμματισμού
- Προσφέρεται για προγραμματισμό σε χαμηλό επίπεδο (κοντά στη γλώσσα μηχανής) και για αυτό συχνά αποκαλείται 'high-level assembly' ή 'portable assembly'
- Χαρακτηρίζεται από πολύ καλή μεταφερτότητα: μεταγλωττιστές για την C υφίστανται σχεδόν για οποιονδήποτε εμπορικό επεξεργαστή
- Η διαδικασία της μεταγλώττισης από την C είναι σχετικά απλή και κατανοητή

Προκαθορισμένοι τύποι δεδομένων στην C

- Βασικοί τύποι δεδομένων: `char`, `int`, `float`, `double`
- Η δήλωση `void` εκφράζει την απουσία τύπου
- Τα προσδιοριστικά `signed`, `unsigned` καθορίζουν το πρόσημο του αριθμού (μόνο για ακεραίους) και τα `short`, `long` το εύρος του σε bit (για `char` και `int`, το `long` μπορεί να χρησιμοποιηθεί και στη δήλωση ενός `double`)
- Οι τύποι `float` και `double` δηλώνουν αριθμούς κινητής υποδιαστολής απλής και διπλής ακρίβειας κατά το πρότυπο IEEE-754
- Το εύρος τιμών των `char`, `int` συνήθως εξαρτάται από το εύρος bit των καταχωρητών του επεξεργαστή και των τρόπων χειρισμού τους. Συχνά: `sizeof(char) = 1 byte`, `sizeof(int) = 4 bytes`
- (Σχεδόν) πάντα: `1 byte = 8 bits`
- Υπάρχουν τρόποι για την υποστήριξη
  - Αριθμητικής αυθαίρετης ακρίβειας με χρήση κατάλληλων βιβλιοθηκών λογισμικού, π.χ. *gmp*
  - Εξομοίωσης της αριθμητικής κινητής υποδιαστολής από ρουτίνες πράξεων μόνο με ακεραίους: *SoftFloat*

Βασικές δομές ελέγχου: Δήλωση if-then-else

- Αναπαράσταση της δήλωσης `if` σε διαγράμματα ροής



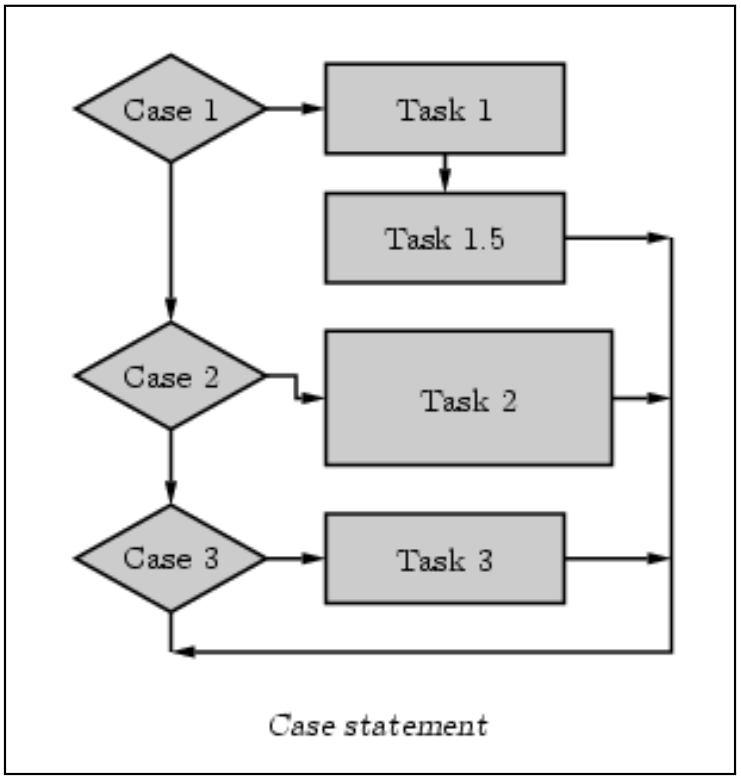
If-then-else
(the True and False tasks are optional)

- Δήλωση `if-else` στην C

```
if (A < B) {
   True;
} else {
   False;
}
```

Βασικές δομές ελέγχου: Δήλωση case

- Αναπαράσταση της δήλωσης `case` σε διαγράμματα ροής



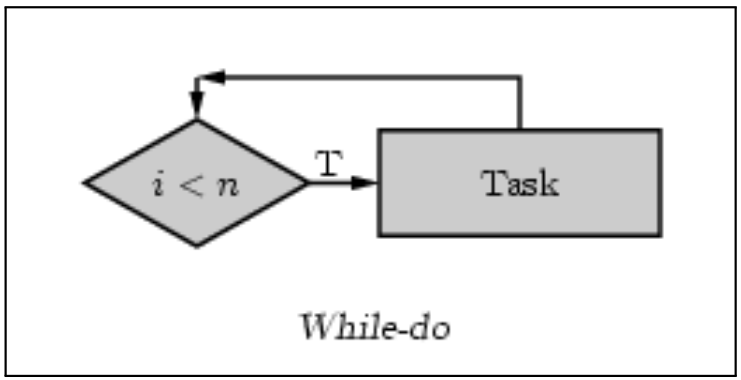
- Δήλωση `switch-case` στην C

```
switch (...) {
  case 1:
    Task 1;
    Task 1.5;
    break;
 case 2:
    Task 2;
    break;
  case 3;
    Task 3;
    break;
}
```

- Χωρίς `break` οι δηλώσεις `case` είναι fall-through
- `default: break;` μετά την τελευταία `case`

Βασικές δομές επανάληψης: Δήλωση while-do

- Έλεγχος συνθήκης πριν την εκτέλεση της πρώτης επανάληψης
- Αναπαράσταση της δήλωσης `while-do` σε διαγράμματα ροής



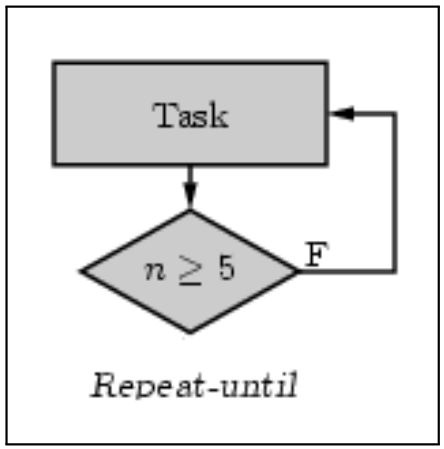
- Δήλωση `while-do` στην C

```
while (i < n) {
  Task;
}
```

- Είναι η θεμελιώδης δομή επανάληψης στην C. Οι δομές `for` και `do-while` μπορούν να εκφραστούν με τη βοήθεια της `while-do`

Βασικές δομές επανάληψης: Δήλωση repeat-until

- Εκτέλεση της πρώτης επανάληψης πριν τον έλεγχο της συνθήκης
- Αναπαράσταση της δήλωσης `repeat-until` σε διαγράμματα ροής

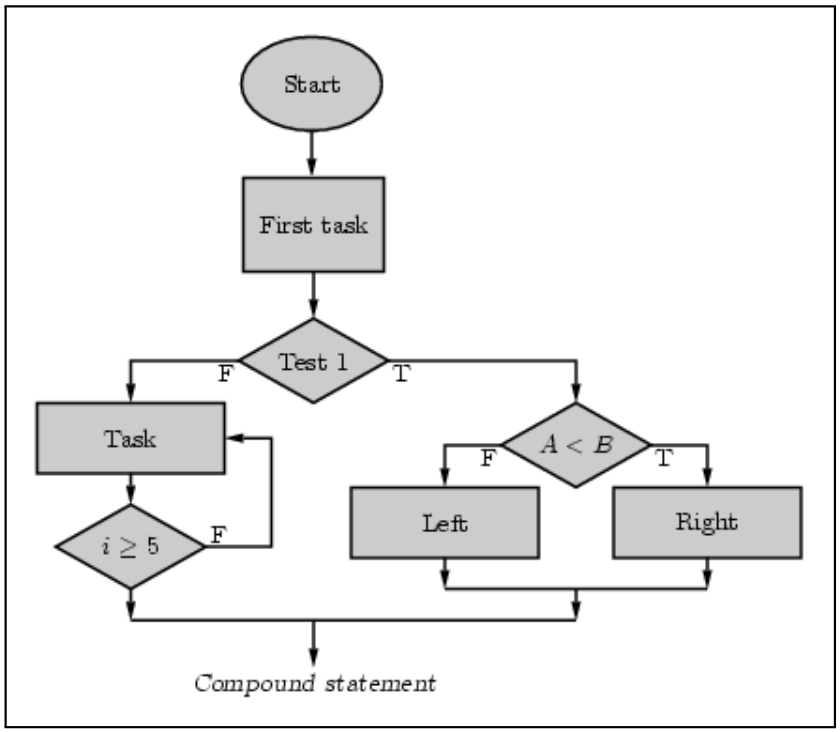


- Αντιστοιχεί στη δήλωση `do-while` της C
- ☞ Στην `do-while` χρησιμοποιείται η **ΣΥΜΠΛΗΡΩΜΑΤΙΚΗ** συνθήκη ελέγχου

```
do {
  Task;
} while (n < 5);
// uses the
// complementary
// condition of
// (n >= 5)
```

Παράδειγμα σύνθετης δομής

- Χρησιμοποιώντας τις βασικές δομές συνθέτουμε πολυπλοκότερες δομές οι οποίες αναπαριστούν αντίστοιχα τμήματα κώδικα στην πηγαία γλώσσα



- Η σύνθετη δήλωση στην C

```
First task;
if (Test 1) {
  if (A < B) {
    Right;
  } else {
    Left;
  }
} else {
  do {
    Task;
  } while (i < 5);
}
```

Δηλώσεις άλματος

- Οι δηλώσεις άλματος (jump statements) στην C είναι: `break`, `continue`, `goto`

ℹ Και οι τρεις δηλώσεις μπορούν να αποφευχθούν όταν συντάσσουμε προγράμματα στην C: δομημένος προγραμματισμός

- `break;`
  - Η `break` προσφέρει πρόωρη έξοδο από τις δηλώσεις `for`, `while`, `do-while`, `switch`
  - Προκαλεί την άμεση έξοδο από τον εσώτερο βρόχο
- `continue;`
  - Χρησιμοποιείται στις δηλώσεις επανάληψης
  - Προκαλεί την άμεση έναρξη της επόμενης επανάληψης (iteration) με αντίστοιχη αύξηση του δείκτη
- `goto label;`
  - Προκαλεί άλμα στη θέση που σημειώνει η ετικέττα *label*
  - GOTO statement considered harmful ...

Πίνακες

- Οι πίνακες στην C αποτελούν συστοιχίες από ομοειδή στοιχεία τα οποία αποθηκεύονται σε διαδοχικές θέσεις στην κεντρική μνήμη του συστήματος
- Δήλωση πίνακα
  `int a[3];`
- Μηδενισμός των στοιχείων ενός πίνακα (απαραίτητος για δυναμικά καταμερισμένη μνήμη διαχειριζόμενη από συναρτήσεις τύπου `malloc()` και `free()`)
  `for (int i=0; i<len; a[i++]=0);`
- Διευθυνσιοδότηση πίνακα και χρήση δεικτών (pointers)
  - Έστω ο δείκτης `int *pa;`
  - Αντιστοίχιση στον πίνακα: `pa = a;`
  - Πρόσβαση στο στοιχείο στη θέση *i*: `a[i]` ή `*(a+i)`

Δήλωση συναρτήσεων (functions) στην C

- Η C υποστηρίζει τη δήλωση συναρτήσεων οι οποίες έχουν προαιρετικά επιστρεφόμενη τιμή και λίστα ορισμάτων
- Σύνταξη μιας συνάρτησης στην C

```
[return-type] function-name ([parameter-list]) {
  declarations
  statements
}
```

- Παράδειγμα συνάρτησης: Αναγωγή σε δύναμη

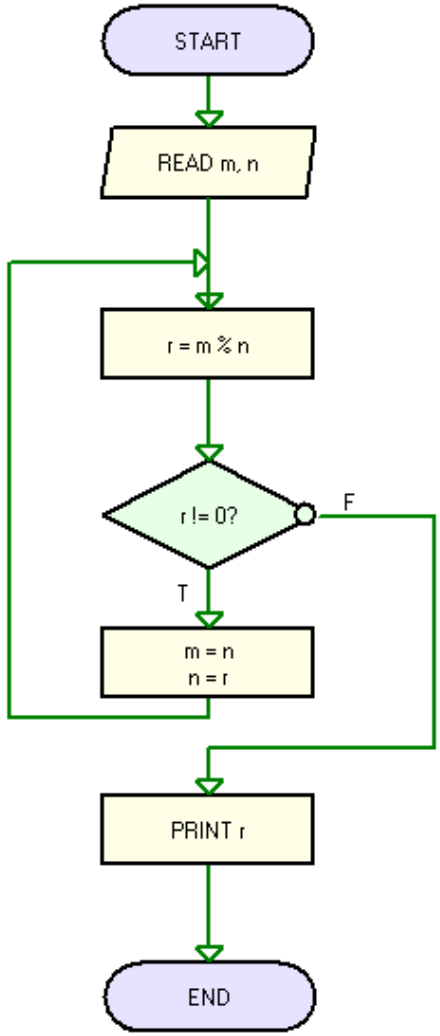
```
int power(int base, int exp)
{
  int i, p = 1;
  for (i=1; i<=exp; i++) {
    p *= base;
  }
  return p;
}
```

Η συνάρτηση `main`

- Η συνάρτηση `main` αποτελεί το σημείο εισόδου για την εκτέλεση ενός προγράμματος στην C
- Η `main` μπορεί να δηλωθεί χωρίς ορίσματα: `int main(void)` στην ISO C
- Η πλήρης δήλωσή της περιλαμβάνει τα ειδικά ορίσματα `argc` και `argv` όπως φαίνεται στο εξής πρότυπο συνάρτησης (function prototype):
  `int main(int argc, char *argv[]);`
- Με τη δήλωση αυτή, η `main` μπορεί να δεχθεί ορίσματα από τη γραμμή εντολών
- Το όρισμα `argc` (=argument counter) είναι ο αριθμός των ορισμάτων που δίνεται ως είσοδος στην εκτελέσιμη μορφή του προγράμματος από τη γραμμή εντολών
- Το όρισμα `argv` (=argument vector) είναι ένας δείκτης στον πίνακα που περιέχει τις συμβολοσειρές στις οποίες καταγράφηκαν τα ορίσματα εισόδου του προγράμματος.

Παράδειγμα πηγαίου προγράμματος σε ANSI C: Αλγόριθμος υπολογισμού του μέγιστου κοινού διαιρέτη

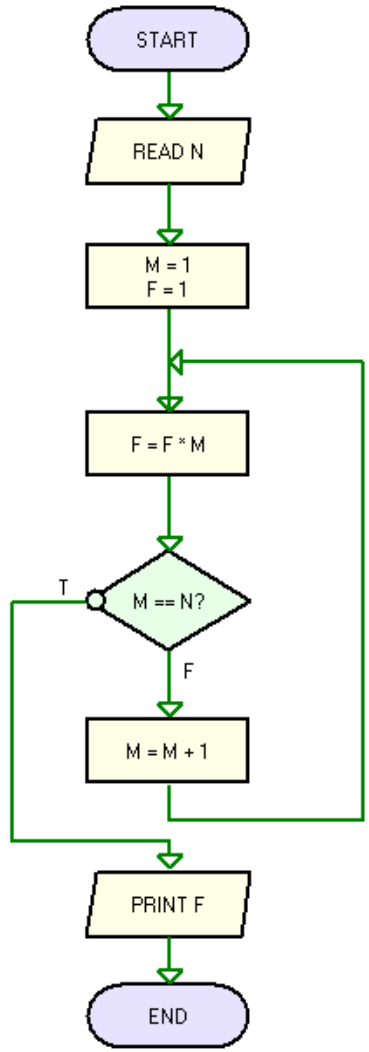
```c
#include <stdio.h>
#include <stdlib.h>

int gcd(int m, int n) {
  int r;
  while ((r = m % n) != 0) {
    m = n;
    n = r;
  }
  return n;
}

// $ ./gcd 60 8
// $ gcd(60,8)=4
void main(int argc, char *argv[]) {
  int a=atoi(argv[1]), b=atoi(argv[2]);
  printf("gcd(%d,%d)=%d\n",
    a, b, gcd(a, b));
}
```



Ένα ακόμη παράδειγμα: Η συνάρτηση factorial

- Υπολογισμός του $N!$

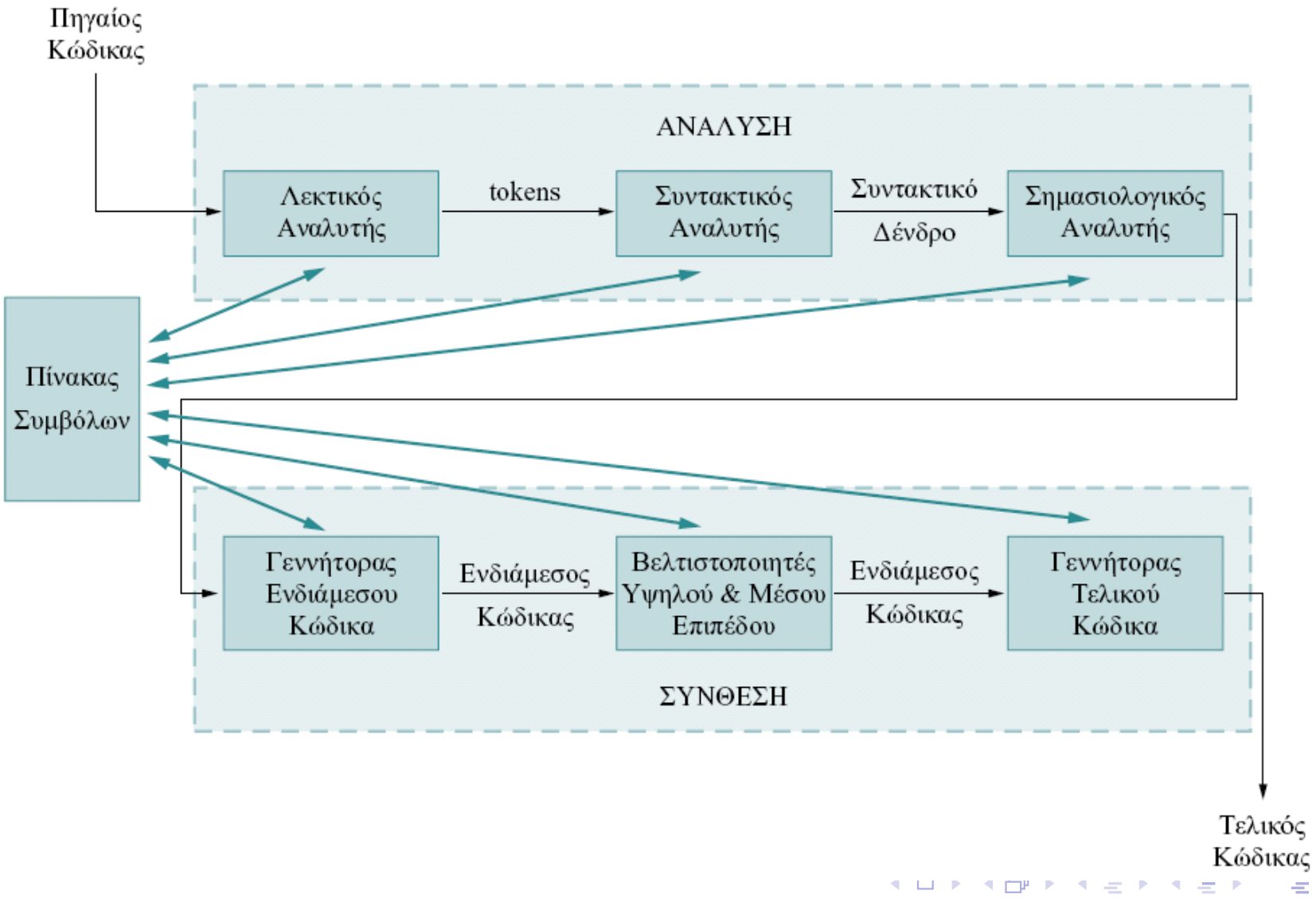
```
// factorial.c
int factorial(int N)
{
  int M, F;

  M = 1;
  F = 1;

  for (M=1; M!=N; M++)
  {
    F = F * M;
  }

  return F;
}
```



Η έννοια της μεταγλώττισης (compilation)

- Η έννοια της μεταγλώττισης στην επιστήμη των υπολογιστών είναι γενική
- Αναφέρεται στη διαδικασία της μετάφρασης (μετατροπής) από μία πηγαία γλώσσα η οποία διέπεται από λεκτικούς και συντακτικούς κανόνες (γραμματική) σε κάποια τελική γλώσσα στο ίδιο ή διαφορετικό επίπεδο αφαίρεσης
- Το τελικό πρόγραμμα διατηρεί σημασιολογική ισοδυναμία με το πηγαίο πρόγραμμα
- Με τον όρο compiler αναφέρεται το λογισμικό που επιτελεί τη μετάφραση από γλώσσα υψηλού επιπέδου (HLL) στο επίπεδο του κώδικα μιας πραγματικής ή εικονικής μηχανής
- Οι πρακτικοί μεταγλωττιστές αποτελούνται από πολλά τμήματα τα οποία εκτελούνται διαδοχικά

Τυπικός σχεδιασμός ενός μεταγλωττιστή ([Aho, 2010, (μετφρ. Ελληνικά), Πιντέλας, 2003])



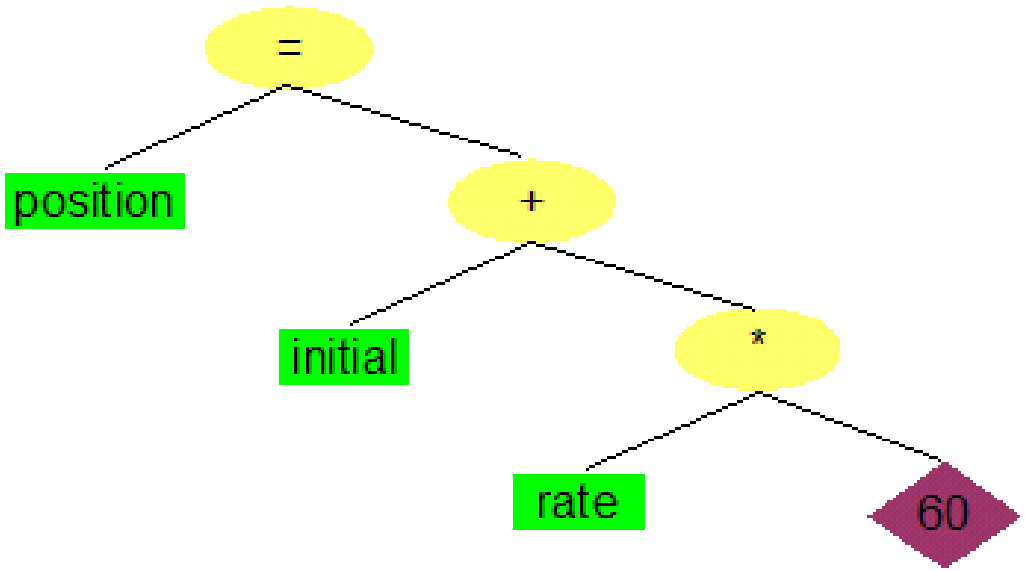
Λεκτικός αναλυτής (lexical analyzer)

- Ο λεκτικός αναλυτής ομαδοποιεί τους χαρακτήρες της εισόδου σε Λεκτικές Μονάδες (tokens)
- Τα πρότυπα των λεκτικών μονάδων εκφράζονται μέσω *κανονικών εκφράσεων*
- Παράδειγμα
  `position = initial + rate*60;`
- Ανάλυση σε λεκτικές μονάδες

`position, initial, rate`	τύπου προσδιοριστή (identifier)
`60`	τύπου σταθεράς
`=, *, +`	τύπου πράξης (τελεστές)
`;`	τύπου στίξης (διαχωριστής εντολών)

Συντακτικός αναλυτής (syntax analyzer)

- Ομαδοποιεί τις λεκτικές μονάδες τις οποίες αναγνωρίζει ο λεκτικός αναλυτής σε συντακτικές μονάδες
- Παράγει αναπαράσταση του πηγαίου προγράμματος σε μορφή συντακτικού δένδρου (ΑΣΤ)
- Γίνεται έλεγχος για την ύπαρξη συντακτικών λαθών εφαρμόζοντας τους κανόνες της γραμματικής της γλώσσας
- Το συντακτικό δένδρο για την έκφραση `position = ...`



Ο σημασιολογικός αναλυτής (semantics analyzer)

- Ο σημασιολογικός αναλυτής αναλύει το συντακτικό δένδρο χρησιμοποιώντας πληροφορία που σχετίζεται τόσο με τους τύπους όσο και με τις τιμές των συμβόλων
- Στο παράδειγμα, έλεγχος των δηλώσεων (declaration) των αναγνωριστικών πριν τη χρησιμοποίηση τους στον κώδικα
- Η σημασιολογική ανάλυση ενημερώνει την αναπαράσταση συντακτικού δένδρου με πληροφορία όπως για την εγκυρότητα χρήσης των τελεστών ανάλογα με τους τύπους των αναγνωριστικών: π.χ. το αναγνωριστικό `rate` είναι τύπου `REAL`
- Σε ορισμένους απλούς μεταγλωττιστές, ο σημασιολογικός αναλυτής μπορεί να παράγει ενδιάμεσο ή τελικό κώδικα
  - Αφορά πολύ απλές μηχανές όπως επεξεργαστές στοίβας
  - Ο παραγόμενος κώδικας δεν είναι βελτιστοποιημένος

Γεννήτορας ενδιάμεσου κώδικα

- Δέχεται ως είσοδο το συντακτικό δένδρο και παράγει κώδικα για μια απλή εικονική (virtual) μηχανή
- Ενδιάμεση αναπαράσταση (IR: Intermediate Representation) όπως μορφής κώδικα τριών διευθύνσεων ή τετράδων (quadruples)
- Για το παράδειγμα:

```
temp1 = int2real(60);
temp2 = rate * temp1;
temp3 = initial + temp2;
position = temp3;
```

- Η χρήση IR επιτρέπει την παραγωγή τελικού κώδικα για διαφορετικές μηχανές (στοχευόμενες αρχιτεκτονικές) χωρίς την επανάληψη της λεκτικής, συντακτικής και σημασιολογικής ανάλυσης του πηγαίου προγράμματος
- Για κάθε νέα αρχιτεκτονική χρειάζεται το αντίστοιχο τμήμα του μεταγλωττιστή για την γέννηση τελικού κώδικα από την IR
- Το τελευταίο ισχύει στους επαναστοχεύσιμους μεταγλωττιστές (retargetable compilers)

Βελτιστοποιητές υψηλού και μεσαίου επιπέδου

- Οι βελτιστοποιητές υψηλού και μεσαίου επιπέδου περιλαμβάνουν πολλές αλληλοδιαδεχόμενες διαδικασίες: αναλύσεις ή/και μετασχηματισμούς
- ☞ Ονομάζονται 'περάσματα' (passes) του μεταγλωττιστή
- Σκοπός είναι η 'βελτίωση' του ενδιάμεσου κώδικα του πηγαίου προγράμματος
- Τελικό ζητούμενο είναι ο ταχύτερα εκτελούμενος τελικός κώδικας και οι μικρότερες απαιτήσεις μνήμης για εντολές και δεδομένα του προγράμματος
- Για το παράδειγμα, το αποτέλεσμα των βελτιστοποιήσεων θα μπορούσε να είναι:

```
temp1 = rate * 60.0;
position = initial + temp1;
```

- Η `temp1` είναι προσωρινή μεταβλητή (temporary variable)
- Οι βελτιστοποιήσεις μεταγλωττιστών αποτελούν ανοικτό πεδίο έρευνας

Γεννήτορας τελικού κώδικα

- Ο γεννήτορας τελικού κώδικα δέχεται ως είσοδο την IR και παράγει ως έξοδο τον τελικό κώδικα (γλώσσα μηχανής ή συμβολομεταφραστή)
  - Ανάθεση των μεταβλητών στις αντίστοιχες θέσεις μνήμης
  - Επιλογή των εντολών επιπέδου συμβολομεταφραστή (εντολές από το ρεπερτόριο του επεξεργαστή ή της εικονικής μηχανής)
  - Αντιστοίχιση μεταβλητών με καταχωρητές του επεξεργαστή
- Τελικός κώδικας για το παράδειγμα (MIPS32)

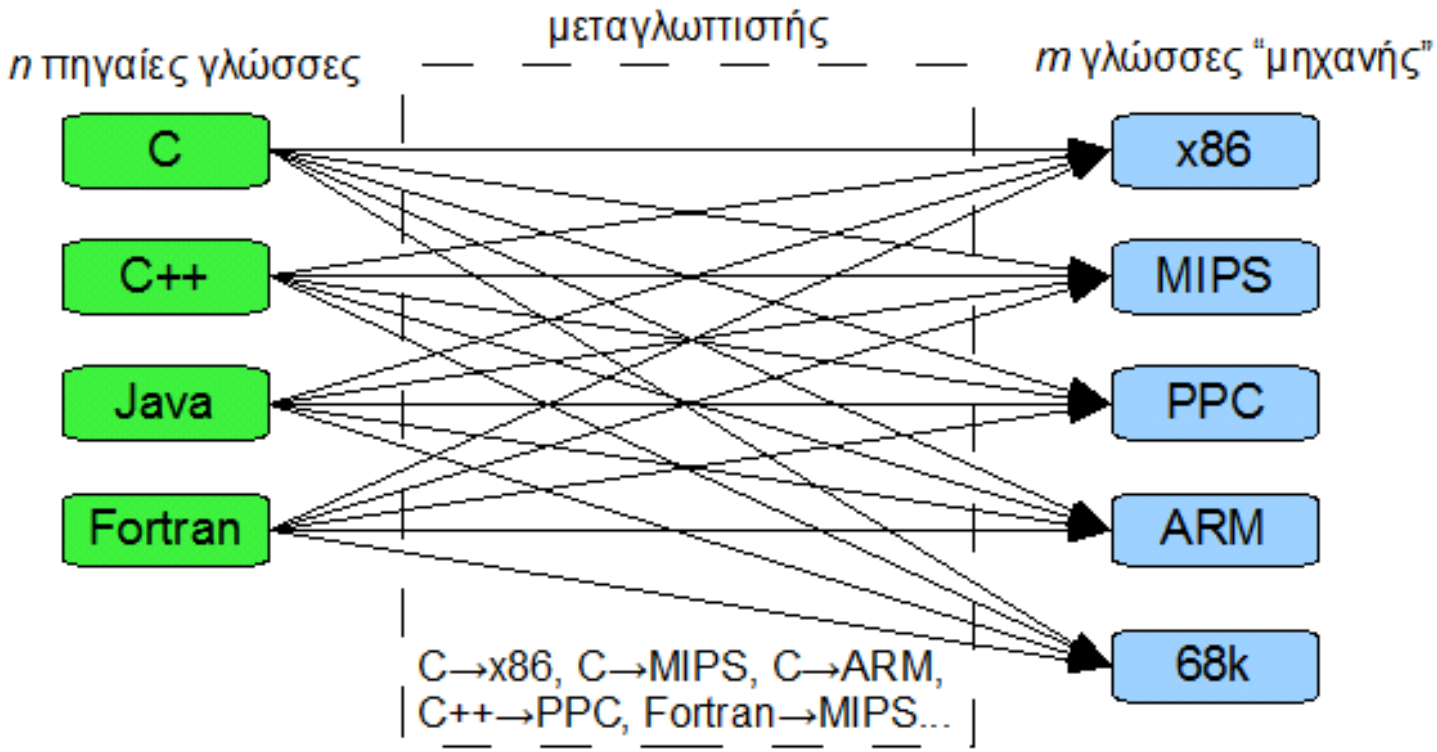
```
mul.s $f2, $f1, 60.0
add.s $f3, $f3, $f2
```

Ο πίνακας συμβόλων (symbol table)

- Ο πίνακας συμβόλων βρίσκεται στην 'καρδιά' του μεταγλωττιστή
- Είναι η βάση δεδομένων για τα σύμβολα του πηγαίου προγράμματος
- Υποβοηθά όλα τα υπόλοιπα τμήματα του μεταγλωττιστή παρέχοντας τις δομές που απαιτούνται, για να τοποθετήσουν και να ανακτήσουν πληροφορίες
- Η εγγραφή μιας μεταβλητής στον πίνακα συμβόλων περιέχει το όνομά της, τη διεύθυνση μνήμης, τον τύπο και την εμβέλειά της
- Μια εγγραφή συνάρτησης θα περιέχει μεταξύ άλλων το όνομά της, τον αριθμό και τον τύπο των ορισμάτων της

Μεταγλωττιστές για πολλές πηγαίες και τελικές γλώσσες

- Αρκετά νωρίς (αρχές ΄50) τέθηκε το πρόβλημα του μεταγλωττιστή ο οποίος κατανοεί πολλές (n) γλώσσες εισόδου και παράγει κώδικα σε (m) γλώσσες χαμηλού επιπέδου
- Ο μεταγλωττιστής αυτός θα έπρεπε να επιτελεί $n \times m$ τρόπους μετάφρασης

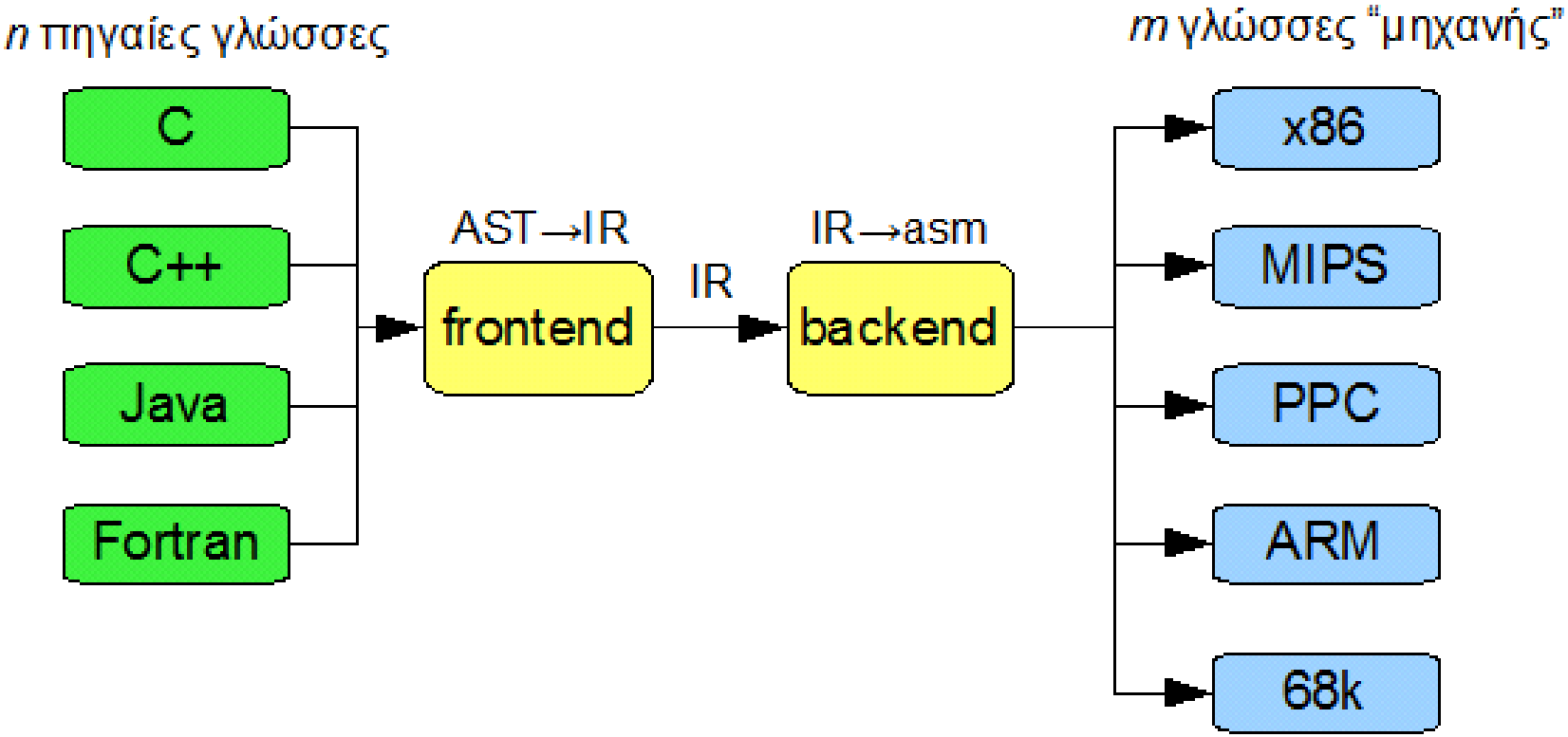


Η χρησιμότητα της ενδιάμεσης αναπαράστασης

- Λόγω της προφανούς μη πρακτικότητας, προτάθηκε διαδικασία δύο σταδίων:
  1. μετάφραση από την πηγαία γλώσσα (HLL: High-Level Language) σε μια ενδιάμεση αναπαράσταση (IR)
  2. μετάφραση από την ενδιάμεση αναπαράσταση στη γλώσσα χαμηλού επιπέδου (LLL: Low-Level Language)
- Υποστήριξη $n + m$ μηχανισμών μεταγλώττισης αντί για $n \times m$
- Το τμήμα του μεταγλωττιστή που επιτελεί την μετάφραση HLL→IR ονομάζεται *frontend*
- Το τμήμα του μεταγλωττιστή που επιτελεί την μετάφραση IR→LLL ονομάζεται *backend*

Ο πρακτικός μεταγλωττιστής

- Με χρήση της IR απαιτούνται 9 μηχανισμοί μεταγλώττισης αντί για 20



Αναφορές του μαθήματος Ι

B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, 2nd ed. Prentice Hall PTR, One Lake Street, Upper Saddle River, New Jersey 07458, USA: Prentice-Hall, 1988.

A. V. Aho, R. Sethi, and J. D. Ullman, *Μεταγλωττιστές: Αρχές, Τεχνικές και Εργαλεία*, με την επιμέλεια των: Αγγελος Σπ. Βώρος και Νικόλαος Σπ. Βώρος και Κων/νος Γ. Μασσέλος, **κεφάλαια 1.1, 3.3**, Εκδόσεις Νέων Τεχνολογιών, 2010. Website for the English version: `http://dragonbook.stanford.edu`

Παναγιώτης Πιντέλας και Παναγιώτης Αλεφραγκής, *Μεταγλωττιστές (Πρακτική Εξάσκηση σε Θέματα Λογισμικού: Τόμος Ά)*, ***κεφάλαια 1, 2.3***. Ελληνικό Ανοικτό Πανεπιστήμιο, 2003. [Online]. Available: `http://www.pli.eap.gr/pdf/PLH40/PLH40-1/pintellas2.pdf`