

Αντικείμενο του μαθήματος Προηγμένα Θέματα Θεωρητικής Πληροφορικής

Προηγμένα Θέματα Θεωρητικής Πληροφορικής Η οργάνωση του μεταγλωττιστή

Νικόλαος Καββαδίας
nkavn@uop.gr

10 Μαρτίου 2010

- Επιμέρους στόχοι του μαθήματος
 - Παρουσίαση θεμάτων που άπτονται του σχεδιασμού μεταγλωττιστών πέρα από τα βασικά
 - Θεωρητικό υπόβαθρο
 - 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 Προηγμένα Θέματα Θεωρητικής Πληροφορικής (2)

- Επιθυμητές (στην πράξη, προαπαιτούμενες) γνώσεις για επιτυχή παρακολούθηση του μαθήματος
 - ✓ Απευθύνεται σε φοιτητές που έχουν διδαχθεί και εξεταστεί επιτυχώς σε μάθημα αντίστοιχο των 'Μεταγλωττιστών Ι' και μαθήματα προγραμματισμού (όπως Ιου και ΙΙου 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

Περιγραφή μαθήματος (1)

- 1 Η οργάνωση του μεταγλωττιστή
- 2 Γέννηση ενδιάμεσης αναπαράστασης
- 3 Επιλογή κώδικα
- 4 Κατανομή καταχωρητών
- 5 Ανάλυση ροής δεδομένων και ελέγχου
- 6 Βελτιστοποιήσεις ανεξάρτητες από την αρχιτεκτονική
- 7 Χρονοπρογραμματισμός κώδικα και παραλληλία επιπέδου εντολών
- 8 Βελτιστοποιήσεις για την ανάδειξη της παραλληλίας εντολών και της τοπικότητας δεδομένων
- 9 Περιβάλλον χρόνου εκτέλεσης
- 10 Γέννηση τελικού κώδικα για αρχιτεκτονικές μηχανής
- 11 Διασυναρτησιακές βελτιστοποιήσεις
- 12 Επαναστοχεύσιμοι μεταγλωττιστές και εργαλεία ανάπτυξης μεταγλωττιστών

Περιγραφή μαθήματος (2)

- Προχωρημένα θέματα στους μεταγλωττιστές
 - Περιβάλλον χρόνου εκτέλεσης
 - Βελτιστοποιήσεις για την εκμετάλλευση της παραλληλίας και ενίσχυση της τοπικότητας
 - Μετασχηματισμοί πηγαίου κώδικα (source-to-source transformations)
 - Επαναστοχεύσιμοι μεταγλωττιστές και εργαλεία ανάπτυξης μεταγλωττιστών
 - Μεταγλωττιστές: GCC, Portable C Compiler (PCC), LCC, LLVM, COINS, Trinaran, SUIF/Machine-SUIF, Microsoft Phoenix
 - Εργαλεία του οικοσυστήματος των μεταγλωττιστών: flex, bison, BURG clones, OLIVE, binutils, newlib, copt, superopt, Graphviz, VCG

Εισαγωγικά

- Στα πλαίσια του μαθήματος θα διδαχθεί το αντικείμενο των προηγμένων θεμάτων στη σχεδίαση μεταφραστών γλωσσών προγραμματισμού
- Το μάθημα θα εστιάσει στη σύνθεση τελικού κώδικα για αρχιτεκτονικές μηχανής (προγραμματιζόμενοι επεξεργαστές) από ενδιάμεση αναπαράσταση του πηγαίου προγράμματος
- 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 ιστότοπο του μαθήματος που θα σας ανακοινωθεί

Κανονικές εκφράσεις (regular expressions)

- Κανονική έκφραση: αλγεβρική σημειογραφία για την περιγραφή συνόλων συμβολοσειρών (strings)
- Αποτελεί ένα είδος αναπαράστασης ένα-σε-πολλά (μία κανονική έκφραση αντιπροσωπεύει πολλές συμβολοσειρές)
- Κανόνες

Σύμβολο	Περιγραφή
?	προαιρετική εμφάνιση (0 ή 1 φορές)
*	0 ή περισσότερες εμφανίσεις
+	1 ή περισσότερες εμφανίσεις
	εναλλαγή (διάζευξη)
\	χαρακτήρας διαφυγής
(...)	ομαδοποίηση
[...]	ορισμός κλάσης χαρακτήρων
-	διάστημα χαρακτήρων (μέσα σε κλάση)
^	σμπλήρωμα του συνόλου χαρακτήρων (μέσα σε κλάση)
.	όλοι οι χαρακτήρες εκτός από τον EOL (End Of Line)
ε	κενή συμβολοσειρά

Παραδείγματα κανονικών εκφράσεων

- Ένα παράδειγμα: $(ab|cd+)?(ef)^*$
 - Ταυτίζεται με: abefef, efefef, cdef, cddd
 - Δεν ταυτίζεται με: abc, abcd, abcdef
- Συχνά χρησιμοποιούμενες κανονικές εκφράσεις

Περιγραφή	Κανονική έκφραση
Πεζός λατινικός χαρακτήρας	[a-z]
Ακέραιος αριθμός	[+]?[0-9]+
Έγκυρα ονόματα αναγνωριστικών στην C	[a-zA-Z_][a-zA-Z_0-9]*
Αριθμοί κινητής υποδιαστολής	[+]?((([0-9]+)(.[0-9]*)?) . [0-9]+)([eE][+]?[0-9]+)?

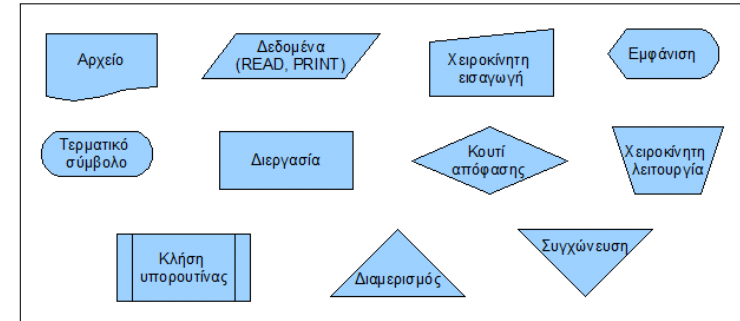
Περιγραφή του συντακτικού μιας πηγαίας γλώσσας

- Η περιγραφή της οργάνωσης μιας πηγαίας γλώσσας γενικά γίνεται με την 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

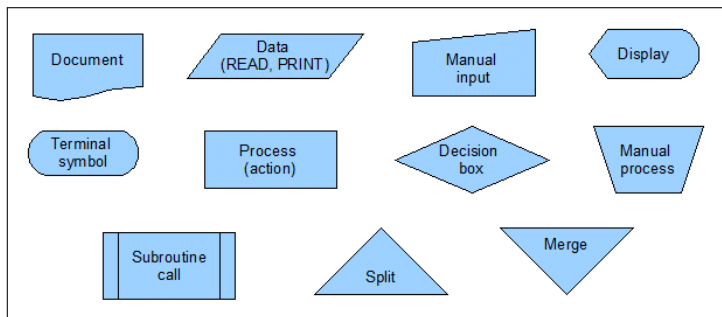
Σχεδιασμός διαγράμματος ροής (flow chart)

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



Σχεδιασμός διαγράμματος ροής (στα Αγγλικά)

- Οι χρησιμοποιούμενες διεθνώς ονομασίες των ενεργειών (προγραμματικών δομών) που μπορούν να αναπαρασταθούν σε ένα flow chart



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 δυνατοτήτων χειρισμού τους
- 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`

Ο τελεστής `sizeof`

- Η C προσφέρει ένα μοναδιαίο τελεστή ο οποίος επιστρέφει το μέγεθος του ορίσματος του σε bytes
- Η απόκρισή του εξαρτάται από τον μεταγλωττιστή της C και χρησιμοποιεί την γνώση του για την αντιστοίχιση των βασικών τύπων δεδομένων με πόρους του επεξεργαστή
- Κανόνες

```
sizeof(char) = sizeof(signed char) = sizeof(unsigned char)
sizeof(char) ≤ sizeof(short int) ≤ sizeof(int) ≤ sizeof(long int)
sizeof(float) ≤ sizeof(double) ≤ sizeof(long double)
```

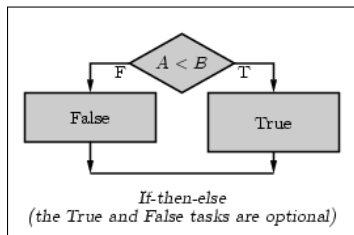
- Παράδειγμα

```
char a; short int b; unsigned int c; long int d;
long long int e; float f; double g; long double h;
printf("Results: %d, %d, %d, %d, %d, %d, %d, %d\n",
sizeof(a), sizeof(b), sizeof(c), sizeof(d),
sizeof(e), sizeof(f), sizeof(g), sizeof(h));
```

- Αποτελέσματα σε έναν desktop x86: 1, 2, 4, 8, 4, 8, 12

Βασικές δομές ελέγχου: Δήλωση `if-then-else`

- Αναπαράσταση της δήλωσης `if` σε διαγράμματα ροής

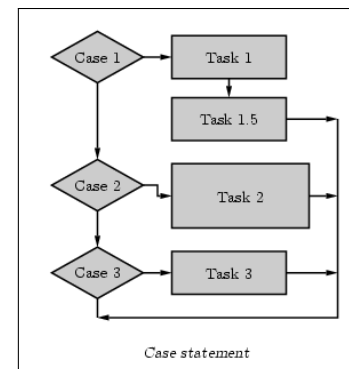


- Δήλωση `if-else` στην C

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

Βασικές δομές ελέγχου: Δήλωση `case`

- Αναπαράσταση της δήλωσης `case` σε διαγράμματα ροής



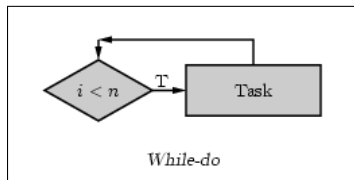
- Δήλωση `switch-case` στην C

```
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



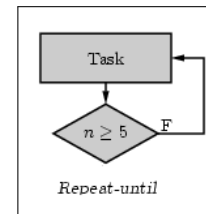
- Δήλωση while-do στην C

```
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

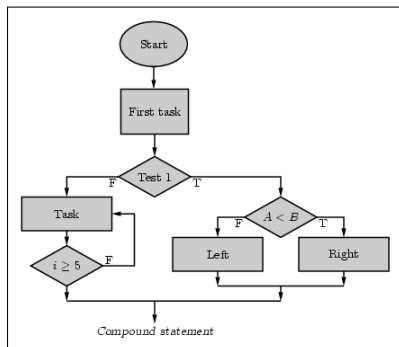


- Αντιστοιχεί στη δήλωση do-while της C
- Στην do-while χρησιμοποιείται η **ΣΥΜΠΛΗΡΩΜΑΤΙΚΗ** συνθήκη ελέγχου

```
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
- Και οι τρεις δηλώσεις μπορούν να αποφευχθούν όταν συντάσσουμε προγράμματα στην C
- break;
 - Η break προσφέρει πρόωγη έξοδο από τις δηλώσεις for, while, do-while, switch
 - Προκαλεί την άμεση έξοδο από τον εσωτερο 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 ετικέτα 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;  
    p = 1;  
    for (i=1; i<=exp; i++)  
        p *= base;  
    return p;  
}
```

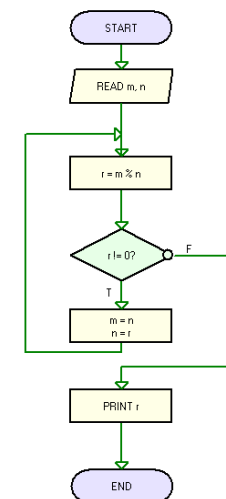
Η συνάρτηση main

- 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 γραμμική εντολών
- Το όρισμα `argc` (=argument counter) είναι ο αριθμός των ορισμάτων που δίνεται ως είσοδος στην εκτελέσιμη μορφή του προγράμματος από τη γραμμική 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

- Ο αλγόριθμος του Ευκλείδη για τον Μέγιστο Κοινό Διαιρέτη

```
// euclid.c  
int gcd(int m, int n)  
{  
    int r;  
    while ((r = m % n) != 0) {  
        m = n;  
        n = r;  
    }  
    return n;  
}  
  
int main()  
{  
    int result = gcd(60, 8);  
    return (result);  
}
```



Ένα ακόμη παράδειγμα: Η συνάρτηση factorial

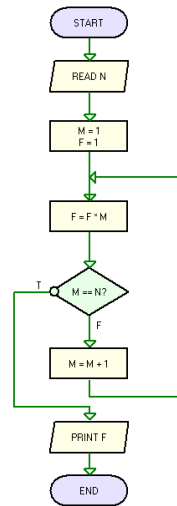
■ Υπολογισμός του $N!$

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

    M = 1;
    F = 1;

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

    return F;
}
```



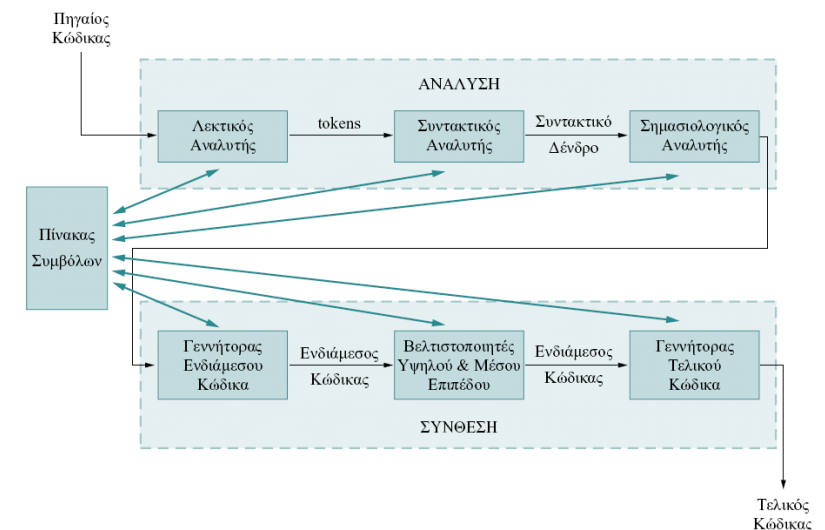
Η έννοια της μεταγλώττισης (compilation)

- 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
- Με τον όρο compiler αναφέρεται το λογισμικό που επιτελεί τη μετάφραση προγραμμάτων σε γλώσσα προγραμματισμού υψηλού επιπέδου (HLL) στο επίπεδο του κώδικα μιας πραγματικής ή εικονικής μηχανής

Η δομή του μεταγλωττιστή

- Η διαδικασία της μεταγλώττισης μπορεί να χωριστεί στην φάση της 'ανάλυσης' και στην φάση της 'σύνθεσης'
 - **ΑΝΑΛΥΣΗ:** αποδόμηση και κατανόηση του πηγαίου προγράμματος
 - **ΣΥΝΘΕΣΗ:** κατασκευή του αποτελέσματος στην τελική γλώσσα διατηρώντας σημασιολογική ισοδυναμία με το πηγαίο πρόγραμμα
- Οι πρακτικοί μεταγλωττιστές αποτελούνται από πολλά τμήματα τα οποία εκτελούνται διαδοχικά
- Στα επιμέρους τμήματα του μεταγλωττιστή μπορεί να πραγματοποιείται ανάλυση και ενδεχόμενα μετασχηματισμός της εισόδου

Τυπικός σχεδιασμός ενός μεταγλωττιστή ([Aho, 2008, (μετφρ. Ελληνικά), Πιντέλας, 2003])



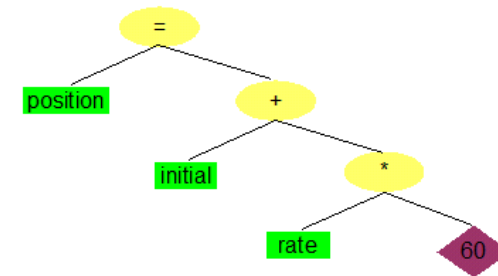
Λεκτικός αναλυτής (lexical analyzer)

- 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

<code>position, initial, rate</code>	τύπου προσδιοριστή (identifier)
<code>60</code>	τύπου σταθεράς
<code>=, *, +</code>	τύπου πράξης (τελεστές)
<code>;</code>	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
- 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 χρησιμοποίησή 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
```

Βελτιστοποιητής χαμηλού επιπέδου

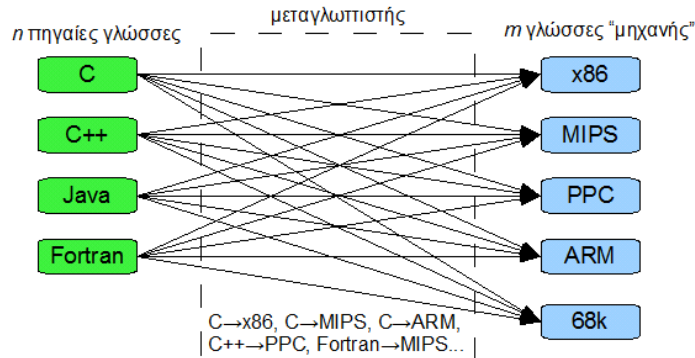
- Ο βελτιστοποιητής χαμηλού επιπέδου χρησιμοποιείται ορισμένες φορές για περαιτέρω βελτίωση του τελικού κώδικα
- Αξιοποιεί ιδιαίτερα χαρακτηριστικά της στοχευόμενης αρχιτεκτονικής
- Παραδείγματα βελτιστοποιητών χαμηλού επιπέδου
 - Χρονοπρογραμματιστής εντολών (instruction scheduler): τοποθετεί τις εντολές του επεξεργαστή σε χρονοθυρίδες (time-slots) για την παράλληλη εκτέλεσή τους
 - Υπερβελτιστοποιητής (superoptimizer): βελτιστοποιεί περιοχές του τελικού κώδικα με εφαρμογή ωμής δύναμης
 - Βελτιστοποιητής κλειδαρότρυπας (peephole optimizer): επιβάλλει μακρο-αντικαταστάσεις με ή χωρίς συνθήκη, εξετάζοντας κάθε φορά ένα 'παράθυρο' του τελικού κώδικα

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 τμήματά του μεταγλωττιστή παρέχοντας τις δομές που απαιτούνται, για να τοποθετήσουν και να ανακτήσουν πληροφορίες
- Μια τυπική εγγραφή στον πίνακα συμβόλων μιας μεταβλητής 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

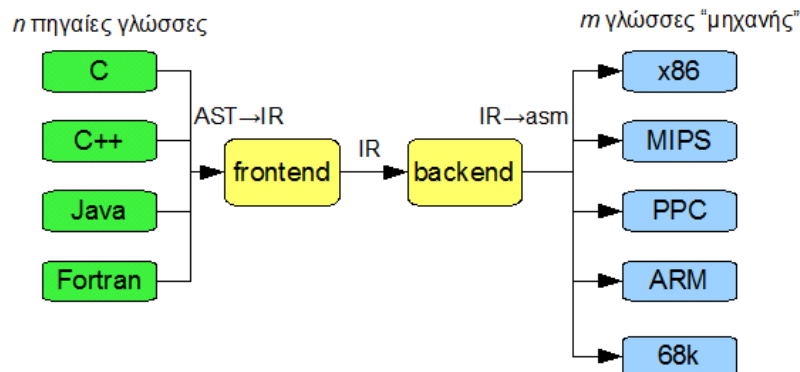


Η χρησιμότητα της ενδιάμεσης αναπαράστασης

- 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



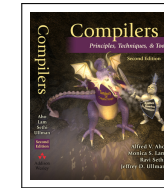
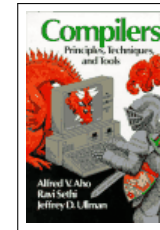
Βασικά εργαλεία στην ανάπτυξη λογισμικού: Ορισμοί ([Δ. Σπινέλλης: Computers for all])

- Κειμενογράφος/Διορθωτής (Editor): Επιτρέπει τη συγγραφή και την αλλαγή του προγράμματος
- Προεπεξεργαστής (Preprocessor): Επεξεργάζεται το πρόγραμμα εκτελώντας απλούς συμβολικούς μετασχηματισμούς και παράγει ένα ισοδύναμο πρόγραμμα (αφορά τις C, C++, Fortran)
- Συμβολομεταφραστής (Assembler): Μετατρέπει τη συμβολική γλώσσα του επεξεργαστή σε γλώσσα μηχανής
- Μεταγλωττιστής (Compiler): Μεταφράζει μια γλώσσα υψηλού επιπέδου σε γλώσσα επιπέδου μηχανής
- Διεγερνευτής (Interpreter): Εκτελεί άμεσα ένα πρόγραμμα σε γλώσσα υψηλού επιπέδου
- Συνδέτης (Linker): Συρράφει τμήματα ενός προγράμματος που έχουν μεταγλωττιστεί ξεχωριστά σε ένα ενιαίο πρόγραμμα
- Φορτωτής (Loader): Φορτώνει το πρόγραμμα στη μνήμη του επεξεργαστή διορθώνοντας αναφορές σε θέσεις μνήμης εντολών και δεδομένων του προγράμματος
- Αποσφαλματωτής (Debugger): Επιτρέπει την εκτέλεση του προγράμματος βήμα-βήμα με σκοπό την ανίχνευση λαθών που μπορεί να περιέχει το πρόγραμμα





Η εργαλειοθήκη του σχεδιαστή μεταγλωττιστών

- Κειμενογράφος/Διορθωτής:
vi, emacs, Geany, Context
- Λεκτικά/συντακτική ανάλυση:
lex+yacc, flex+bison, ANTLR (πρώην PCCTS), GOLD Parser Builder
- Συμβολομεταφραστής-Συνδέτης-Αποσυμβολομεταφραστής:
binutils (as, ld, objdump)
- Μεταγλωττιστής (Compiler):
GCC, LCC, LLVM, COINS, Phoenix, PCC, Trimaran, SUIF/Machine-SUIF, ACSE, xcc, jackcc
- Πρότυπη βιβλιοθήκη της C:
glibc, newlib, uclibc
- Αποσφαλματωτής (Debugger):
GDB
- Γεννίτορες γεννητόρων κώδικα:
BURG, IBURG, LBURG, OLIVE
- Οπτικοποίηση γραφών:
Graphviz, VCG
- Άλλα εργαλεία:
sparse, Aha!, superopt, copt

Χρήσιμα βιβλία



Αναφορές του μαθήματος I

-  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, 2008. Website for the English version: <http://dragonbook.stanford.edu>
-  Παναγιώτης Πιντέλας και Παναγιώτης Αλεφραγκίς, *Μεταγλωττιστές (Πρακτική Εξάσκηση σε Θέματα Λογισμικού: Τόμος Α)*, **κεφάλαια 1, 2.3**. Ελληνικό Ανοικτό Πανεπιστήμιο, 2003. [Online]. Available: <http://www.pli.eap.gr/pdf/PLH40/PLH40-1/pintellas2.pdf>
-  Διομήδης Σπινέλλης. Σειρά διαλέξεων: "Computers For All". [Online]. Available: <http://www.dmst.aueb.gr/dds/cfa/>