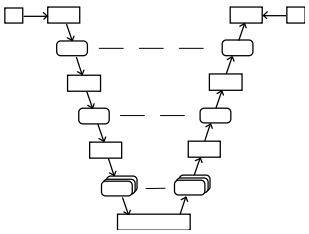
Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών
& Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής & Υπολογιστών
Εργαστήριο Τεχνολογίας Λογισμικού

Μεταγλωττιστές 2024

Θέμα εργασίας

Η γλώσσα Alan

Alan Mathison Turing (1912–1954)



Μεταγλωττιστές

`https://courses.softlab.ntua.gr/compilers/`

Διδάσκων: Κωστής Σαγώνας

Αθήνα, Φεβρουάριος 2024

ΘΕΜΑ:

Να σχεδιαστεί και να υλοποιηθεί από κάθε ομάδα, το πολύ δυο σπουδαστών, ένας μεταγλωττιστής για τη γλώσσα Alan. Γλώσσα υλοποίησης μπορεί να είναι μια από τις C/C++, Rust, Java, SML, OCaml, Haskell ή Python, αλλά έχετε υπ' όψη σας ότι κομμάτια του μεταγλωττιστή και εργαλεία που μπορεί να σας φανούν χρήσιμα μπορεί να μην είναι διαθέσιμα σε κάποιες από τις παραπάνω γλώσσες και σε αυτήν την περίπτωση θα πρέπει να τα υλοποιήσετε μόνοι σας. Η επιλογή κάποιας άλλης γλώσσας υλοποίησης μπορεί να γίνει κατόπιν συνεννόησης με τον διδάσκοντα. Επιτρέπεται να χρησιμοποιηθούν και επίσης συνιστάται η χρήση εργαλείων, π.χ. `flex`/`ML-Lex`/`ocamllex`/`Alex`, `bison`/`ML-Yacc`/`ocamlyacc`/`Happy`, `JavaCC`, κ.λπ., όπως και το LLVM. Περισσότερες πληροφορίες σχετικές με αυτά τα εργαλεία θα δοθούν στις παραδόσεις.

Παραδοτέα, ημερομηνίες και βαθμολόγηση

Τα τμήματα του μεταγλωττιστή και η κατανομή μονάδων φαίνονται στον παρακάτω πίνακα. Οι ημερομηνίες παράδοσης αναγράφονται στη σελίδα του μαθήματος.

Τμήμα του μεταγλωττιστή	*Μονάδες*	*Bonus*
Λεκτικός αναλυτής	0.5	–
Συντακτικός αναλυτής	0.5	–
Σημασιολογικός αναλυτής	1.0	0.5
Ενδιάμεσος κώδικας	1.0	–
Βελτιστοποίηση	1.0	0.8
Τελικός κώδικας	1.0	0.7
Συνολική εργασία και έκθεση	5.0	2.0

Για τα διάφορα τμήματα της εργασίας πρέπει να παραδίδεται εμπρόθεσμα από κάθε ομάδα ο αντίστοιχος κώδικας σε ηλεκτρονική μορφή, καθώς και σαφείς οδηγίες για την παραγωγή ενός εκτελέσιμου προγράμματος επίδειξης της λειτουργίας του αντίστοιχου τμήματος, από τον κώδικα αυτόν. Καθυστερημένες ασκήσεις θα βαθμολογούνται με μικρότερο βαθμό, αντιστρόφως ανάλογα προς το χρόνο καθυστέρησης. Παρακαλούμε, **μην παραδίδετε τυπωμένες εργασίες**! Η μορφή και τα περιεχόμενα των παραδοτέων, συμπεριλαμβανομένης και της τελικής έκθεσης, πρέπει να συμφωνούν με τις οδηγίες που δίνονται στην Ενότητα 4 του παρόντος.

Προαιρετικά τμήματα και μονάδες bonus

Το σύνολο των μονάδων της παρούσας εργασίας είναι 7. Από αυτές, οι 2 μονάδες είναι bonus (που σημαίνει ότι το σύνολο μονάδων του μαθήματος είναι 12) και αντιστοιχούν σε τμήματα της εργασίας που είναι προαιρετικά. Συγκεκριμένα:

- (50%) Βελτιστοποίηση ενδιάμεσου κώδικα με ανάλυση ροής δεδομένων και ελέγχου.
- (50%) Δέσμευση καταχωρητών και βελτιστοποίηση τελικού κώδικα.

Περιεχόμενα

1 Περιγραφή της γλώσσας Alan

Η γλώσσα Alan είναι μια απλή γλώσσα προστακτικού προγραμματισμού. Τα κύρια χαρακτηριστικά της εν συντομία είναι τα εξής:

- Απλή δομή και σύνταξη εντολών και εκφράσεων.
- Βασικοί τύποι δεδομένων για ακέραιους αριθμούς δύο μεγεθών και μονοδιάστατοι πίνακες.
- Απλές συναρτήσεις, πέρασμα κατ' αξία ή κατ' αναφορά.
- Εμβέλεια μεταβλητών όπως στην Pascal.
- Βιβλιοθήκη συναρτήσεων.

Περισσότερες λεπτομέρειες της γλώσσας δίνονται στις παραγράφους που ακολουθούν.

1.1 Λεκτικές μονάδες

Οι λεκτικές μονάδες της γλώσσας Alan χωρίζονται στις παρακάτω κατηγορίες:

- Τις *λέξεις κλειδιά*, οι οποίες είναι οι παρακάτω:

```
byte      else      false     if     int     proc     reference
return    while     true
```

- Τα *ονόματα*, τα οποία αποτελούνται από ένα γράμμα του λατινικού αλφαβήτου, πιθανώς ακολουθούμενο από μια σειρά γραμμάτων, δεκαδικών ψηφίων ή χαρακτήρων υπογράμμισης (underscore). Τα ονόματα δεν πρέπει να συμπίπτουν με τις λέξεις κλειδιά που αναφέρθηκαν παραπάνω.

- Τις *ακέραιες σταθερές* χωρίς πρόσημο, που αποτελούνται από ένα ή περισσότερα δεκαδικά ψηφία. Παραδείγματα ακέραιων σταθερών είναι τα ακόλουθα:

```
0     42     1284     00200
```

- Τους *σταθερούς χαρακτήρες*, που αποτελούνται από ένα χαρακτήρα μέσα σε απλά εισαγωγικά. Ο χαρακτήρας αυτός μπορεί να είναι οποιοσδήποτε κοινός χαρακτήρας ή *ακολουθία διαφυγής* (escape sequence). Κοινοί χαρακτήρες είναι όλοι οι εκτυπώσιμοι χαρακτήρες πλην των απλών και διπλών εισαγωγικών και του χαρακτήρα \ (backslash). Οι ακολουθίες διαφυγής ξεκινούν με το χαρακτήρα \ (backslash) και περιγράφονται στον Πίνακα 1. Παραδείγματα σταθερών χαρακτήρων είναι οι ακόλουθες:

```
'a'     '1'     '\n'     '\"'     '\x1d'
```

- Τις *σταθερές συμβολοσειρές*, που αποτελούνται από μια ακολουθία κοινών χαρακτήρων ή ακολουθιών διαφυγής μέσα σε διπλά εισαγωγικά. Οι συμβολοσειρές δεν μπορούν να εκτείνονται σε περισσότερες από μια γραμμές προγράμματος. Παραδείγματα σταθερών συμβολοσειρών είναι οι ακόλουθες:

```
"abc"          "Route66"          "Helloworld!\n"
     "Name:\t\"DouglasAdams\"\nValue:\t42\n"
```

- Τους *συμβολικούς τελεστές*, οι οποίοι είναι οι παρακάτω:

```
=      +      -      *      /      %      !      &      |
==     !=     <      >      <=     >=
```

Πίνακας 1: Ακολουθίες διαφυγής (escape sequences).

Ακολουθία διαφυγής	Περιγραφή
\n	ο χαρακτήρας αλλαγής γραμμής (line feed)
\t	ο χαρακτήρας στηλοθέτησης (tab)
\r	ο χαρακτήρας επιστροφής στην αρχή της γραμμής (carriage return)
\0	ο χαρακτήρας με ASCII κωδικό 0
\\	ο χαρακτήρας \ (backslash)
\'	ο χαρακτήρας ' (απλό εισαγωγικό)
\"	ο χαρακτήρας " (διπλό εισαγωγικό)
\x$nn$	ο χαρακτήρας με ASCII κωδικό nn στο δεκαεξαδικό σύστημα

- Τους *διαχωριστές*, οι οποίοι είναι οι παρακάτω:

```
(   )   [   ]   {   }   ,   :   ;
```

Εκτός από τις λεκτικές μονάδες που προαναφέρθηκαν, ένα πρόγραμμα Alan μπορεί επίσης να περιέχει τα παρακάτω, τα οποία διαχωρίζουν λεκτικές μονάδες και αγνοούνται:

- *Κενούς χαρακτήρες*, δηλαδή ακολουθίες αποτελούμενες από κενά διαστήματα (space), χαρακτήρες στηλοθέτησης (tab), χαρακτήρες αλλαγής γραμμής (line feed) ή χαρακτήρες επιστροφής στην αρχή της γραμμής (carriage return).

- *Σχόλια μιας γραμμής*, τα οποία αρχίζουν με την ακολουθία χαρακτήρων `--` και τερματίζονται με το τέλος της τρέχουσας γραμμής.

- *Σχόλια πολλών γραμμών*, τα οποία αρχίζουν με την ακολουθία χαρακτήρων `(*` και τερματίζονται με την ακολουθία χαρακτήρων `*)`. Τα σχόλια αυτής της μορφής επιτρέπεται να είναι φωλιασμένα.

1.2 Τύποι δεδομένων

Η Alan υποστηρίζει δύο βασικούς τύπους δεδομένων:

- `int`: ακέραιοι αριθμοί μεγέθους τουλάχιστον 16 bit (-32768 έως 32767), και

- `byte`: μη αρνητικοί ακέραιοι αριθμοί των 8 bit (0 έως 255).

Εκτός από τους βασικούς τύπους, η Alan υποστηρίζει επίσης τύπους πινάκων μιας διάστασης, οι οποίοι συμβολίζονται με t `[]`, όπου ο t πρέπει να είναι κάποιος βασικός τύπος. Υποστηρίζει επίσης έμμεσα έναν τύπο λογικών εκφράσεων, ο οποίος όμως χρησιμοποιείται μόνο στις συνθήκες των εντολών `if` και `while`. Δεν επιτρέπεται η δήλωση μεταβλητών λογικού τύπου, ούτε υπάρχει ανάγκη αναπαράστασης λογικών τιμών στη μνήμη.

1.3 Δομή του προγράμματος

Η γλώσσα Alan είναι μια δομημένη (block structured) γλώσσα. Ένα πρόγραμμα έχει χοντρικά την ίδια δομή με ένα πρόγραμμα Pascal. Οι συναρτήσεις μπορούν να είναι φωλιασμένες η μία μέσα στην άλλη και οι κανόνες εμβέλειας είναι οι ίδιοι με αυτούς της Pascal.

1.3.1 Μεταβλητές

Οι δηλώσεις μεταβλητών γίνονται με αναγραφή του ονόματος και του τύπου της μεταβλητής. Στην περίπτωση που ο τύπος της μεταβλητής είναι τύπος πίνακα, ανάμεσα στις αγκύλες πρέπει να αναγράφεται το μέγεθος του πίνακα, το οποίο πρέπει να είναι θετική ακέραια σταθερά. Παραδείγματα δηλώσεων μεταβλητών είναι:

```
i : int;
b : byte;
s : byte [20];
```

Ας σημειωθεί ότι δεν είναι δυνατή η ταυτόχρονη δήλωση περισσότερων μεταβλητών με τον ίδιο τύπο.

1.3.2 Συναρτήσεις

Κάθε συνάρτηση είναι μια δομική μονάδα και αποτελείται από την επικεφαλίδα της, τις τοπικές δηλώσεις και το σώμα της. Στην επικεφαλίδα αναφέρεται το όνομα της συνάρτησης, οι τυπικές της παράμετροι μέσα σε παρενθέσεις και ο τύπος επιστροφής. Οι παρενθέσεις είναι υποχρεωτικές ακόμα και αν μια συνάρτηση δεν έχει τυπικές παραμέτρους. Ο τύπος επιστροφής δεν μπορεί να είναι τύπος πίνακα. Ο ειδικός τύπος επιστροφής `proc` χρησιμοποιείται για να δηλώσει ότι μια συνάρτηση δεν επιστρέφει αποτέλεσμα (όπως το `void` στη C).

Κάθε τυπική παράμετρος χαρακτηρίζεται από το όνομά της, τον τύπο της και τον τρόπο περάσματος. Η γλώσσα Alan υποστηρίζει πέρασμα παραμέτρων κατ' αξία (by value) και κατ' αναφορά (by reference). Αν στη δήλωση μιας τυπικής παραμέτρου προηγείται του τύπου η λέξη κλειδί `reference`, τότε αυτή περνά κατ' αναφορά, διαφορετικά περνά κατ' αξία. Απαγορεύεται το πέρασμα παραμέτρων τύπου πίνακα κατ' αξία.

Ακολουθούν παραδείγματα επικεφαλίδων συναρτήσεων.

```
p1 () : proc
p2 (n : int) : proc
p3 (a : int, b : int, b : reference byte) : proc
f1 (int x) : int
f2 (s : reference byte []) : int
```

Οι τοπικές δηλώσεις μιας συνάρτησης ακολουθούν την επικεφαλίδα. Η Alan ακολουθεί τους κανόνες εμβέλειας της Pascal, όσον αφορά στην ορατότητα των ονομάτων μεταβλητών, συναρτήσεων και παραμέτρων.

Το σώμα μιας συνάρτησης είναι μια σειρά μηδέν ή περισσότερων εντολών που περικλείεται μέσα σε άγκιστρα `{` και `}`.

1.4 Εκφράσεις και συνθήκες

Κάθε έκφραση της Alan διαθέτει ένα μοναδικό τύπο και μπορεί να αποτιμηθεί δίνοντας ως αποτέλεσμα μια τιμή αυτού του τύπου. Οι εκφράσεις διακρίνονται σε δύο κατηγορίες: αυτές που δίνουν l-values, οι οποίες περιγράφονται στην Ενότητα 1.4.1 και αυτές που δίνουν r-values, που περιγράφονται στις Ενότητες 1.4.2 ως 1.4.4. Τα δυο είδη τιμών l-value και r-value έχουν πάρει το όνομά τους από τη θέση τους σε μια εντολή ανάθεσης: οι l-values εμφανίζονται στο αριστερό μέλος της ανάθεσης ενώ οι r-values στο δεξιό. Εν αντιθέσει με τις l-values, οι r-values δεν μπορούν να έχουν τύπο πίνακα.

Οι συνθήκες της Alan περιγράφονται στις Ενότητες 1.4.2 και 1.4.3. Χρησιμοποιούνται μόνο σε συνδυασμό με τις εντολές `if` και `while` και ο υπολογισμός τους δίνει ως αποτέλεσμα μια λογική τιμή (αληθή ή ψευδή).

Οι εκφράσεις και οι συνθήκες μπορούν να εμφανίζονται μέσα σε παρενθέσεις, που χρησιμοποιούνται για λόγους ομαδοποίησης.

1.4.1 L-values

Οι l-values αντιπροσωπεύουν αντικείμενα που καταλαμβάνουν χώρο στη μνήμη του υπολογιστή κατά την εκτέλεση του προγράμματος και τα οποία μπορούν να περιέχουν τιμές. Τέτοια αντικείμενα είναι οι μεταβλητές, οι παράμετροι των συναρτήσεων, τα στοιχεία πινάκων και οι σταθερές συμβολοσειρές. Συγκεκριμένα:

- Το όνομα μιας μεταβλητής ή μιας παραμέτρου συνάρτησης είναι l-value και αντιστοιχεί στο εν λόγω αντικείμενο. Ο τύπος της l-value είναι ο τύπος του αντίστοιχου αντικειμένου.
- Αν a είναι μια έκφραση τύπου t `[]` και i μια έκφραση τύπου `int`, τότε a`[`i`]` είναι μια l-value με τύπο t, που αντιστοιχεί στο i-οστό στοιχείο του πίνακα a.
- Μια σταθερή συμβολοσειρά, όπως περιγράφεται στην Ενότητα 1.1, είναι l-value με τύπο `byte[`$n+1$`]`, όπου n το μήκος της συμβολοσειράς. Κάθε τέτοια l-value είναι ένας πίνακας αντικείμενων τύπου `byte`, σε κάθε ένα από τα οποία βρίσκονται αποθηκευμένοι με τη σειρά οι ASCII κωδικοί των χαρακτήρων της συμβολοσειράς. Στο τέλος του πίνακα αποθηκεύεται αυτόματα ο αριθμός `0`, σύμφωνα με τη σύμβαση που ακολουθεί η γλώσσα C για τις συμβολοσειρές.

Αν μια l-value χρησιμοποιηθεί ως έκφραση, η τιμή αυτής της έκφρασης είναι ίση με την τιμή που περιέχεται στο αντικείμενο που αντιστοιχεί στην l-value.

1.4.2 Σταθερές

Στις r-values της γλώσσας Alan συγκαταλέγονται οι ακόλουθες σταθερές:

- Οι ακέραιες σταθερές χωρίς πρόσημο, όπως περιγράφονται στην Ενότητα 1.1. Έχουν τύπο `int` και η τιμή τους είναι ίση με τον μη αρνητικό ακέραιο αριθμό που παριστάνουν.
- Οι σταθεροί χαρακτήρες, όπως περιγράφονται στην Ενότητα 1.1. Έχουν τύπο `byte` και η τιμή τους είναι ίση με τον κωδικό ASCII του χαρακτήρα που παριστάνουν.

Επίσης, στις συνθήκες της Alan συγκαταλέγονται οι ακόλουθες σταθερές:

- Οι λέξεις κλειδιά `true` και `false`.

1.4.3 Τελεστές

Οι τελεστές της Alan διακρίνονται σε τελεστές με ένα ή δύο τελούμενα. Οι τελεστές με ένα τελούμενο γράφονται πρίν από αυτό (postfix), ενώ οι τελεστές με δύο τελούμενα γράφονται πάντα μεταξύ των τελουμένων (infix). Η αποτίμηση των τελουμένων γίνεται από αριστερά προς τα δεξιά.

Όλοι οι τελεστές της Alan έχουν ως αποτέλεσμα r-value ή συνθήκη. Περιγράφονται εκτενώς παρακάτω.

- Οι τελεστές με ένα τελούμενο `+` και `-` υλοποιούν τους τελεστές προσήμου. Το τελούμενο πρέπει να είναι έκφραση τύπου `int` και το αποτέλεσμα είναι r-value του ίδιου τύπου.
- Ο τελεστής με ένα τελούμενο `!` υλοποιεί τη λογική άρνηση. Το τελούμενό του πρέπει να είναι συνθήκη και το ίδιο είναι το αποτέλεσμά του.
- Οι τελεστές με δύο τελούμενα `+`, `-`, `*`, `/` και `%` υλοποιούν τις αριθμητικές πράξεις. Τα τελούμενα πρέπει να είναι εκφράσεις του ίδιου τύπου `int` ή `byte` και το αποτέλεσμα είναι r-value του ίδιου τύπου.
- Οι τελεστές `==`, `!=`, `<`, `>`, `<=` και `>=` υλοποιούν τις σχέσεις σύγκρισης μεταξύ αριθμών. Τα τελούμενα πρέπει να είναι εκφράσεις του ίδιου τύπου `int` ή `byte` και το αποτέλεσμα είναι συνθήκη.

Πίνακας 2: Προτεραιότητα και προσεταιριστικότητα των τελεστών της Alan.

Τελεστές	Περιγραφή	Αριθμός τελουμένων	Θέση και προσεταιριστικότητα
`+ - !`	Πρόσημα, λογική άρνηση	1	prefix
`* / %`	Πολλαπλασιαστικοί τελεστές	2	infix, αριστερή
`+ -`	Προσθετικοί τελεστές	2	infix, αριστερή
`== != > < <= >=`	Σχεσιακοί τελεστές	2	infix, καμία
`&`	Λογική σύζευξη	2	infix, αριστερή
`|`	Λογική διάζευξη	2	infix, αριστερή

- Οι τελεστές `&` και `|` υλοποιούν αντίστοιχα τις πράξεις της λογικής σύζευξης και διάζευξης. Τα τελούμενα πρέπει να είναι συνθήκες και το ίδιο είναι και το αποτέλεσμα. Η αποτίμηση συνθηκών που χρησιμοποιούν αυτούς τους τελεστές γίνεται με *βραχυκύκλωση* (short-circuit). Δηλαδή, αν το αποτέλεσμα της συνθήκης είναι γνωστό από την αποτίμηση και μόνο του πρώτου τελούμενου, το δεύτερο τελούμενο δεν αποτιμάται καθόλου.

Στον Πίνακα 2 ορίζεται η προτεραιότητα και η προσεταιριστικότητα των τελεστών της Alan. Οι γραμμές που βρίσκονται υψηλότερα στον πίνακα περιέχουν τελεστές μεγαλύτερης προτεραιότητας. Τελεστές που βρίσκονται στην ίδια γραμμή έχουν την ίδια προτεραιότητα.

1.4.4 Κλήση συναρτήσεων

Αν f είναι το όνομα μιας συνάρτησης με αποτέλεσμα τύπου t, όπου ο τύπος επιστροφής t δεν είναι `proc`, τότε η έκφραση $f(e_1, \ldots, e_n)$ είναι μια r-value με τύπο t. Ο αριθμός των πραγματικών παραμέτρων n πρέπει να συμπίπτει με τον αριθμό των τυπικών παραμέτρων της f. Επίσης, ο τύπος και το είδος κάθε πραγματικής παραμέτρου πρέπει να συμπίπτει με τον τύπο και τον τρόπο περάσματος της αντίστοιχης τυπικής παραμέτρου, σύμφωνα με τους παρακάτω κανόνες.

- Αν η τυπική παράμετρος είναι τύπου t και περνά κατ' αξία, τότε η αντίστοιχη πραγματική παράμετρος πρέπει να είναι έκφραση τύπου t.

- Αν η τυπική παράμετρος είναι τύπου t και περνά κατ' αναφορά, τότε η αντίστοιχη πραγματική παράμετρος πρέπει να είναι l-value τύπου t.

Κατά την κλήση μιας συνάρτησης, οι πραγματικές παράμετροι αποτιμώνται από αριστερά προς τα δεξιά.

1.5 Εντολές

Οι εντολές που υποστηρίζει η γλώσσα Alan είναι οι ακόλουθες:

- Η κενή εντολή `;` που δεν κάνει καμία ενέργεια.

- Η εντολή ανάθεσης ℓ `=` e`;` που αναθέτει την τιμή της έκφρασης e στην l-value ℓ. Η l-value ℓ πρέπει να είναι τύπου t που δεν είναι τύπος πίνακα, ενώ ή έκφραση e πρέπει να είναι του ίδιου τύπου t.

- Η σύνθετη εντολή, που αποτελείται από μια σειρά έγκυρων εντολών ανάμεσα σε άγκιστρα `{` και `}`. Οι εντολές αυτές εκτελούνται διαδοχικά, εκτός αν κάποια είναι εντολή άλματος.

- Η εντολή κλήσης συνάρτησης $f(e_1, \ldots, e_n)$`;`, με τις ίδιες προϋποθέσεις όπως στην Ενότητα 1.4.4 με τη διαφορά ότι ο τύπος επιστροφής της συνάρτησης f πρέπει να είναι `proc`.

- Η εντολή ελέγχου `if (`c`)` s_1 `else` s_2, όπου c πρέπει να είναι μια έγκυρη συνθήκη και s_1, s_2 να είναι έγκυρες εντολές. Το τμήμα `else` είναι προαιρετικό. Η σημασιολογία αυτής της εντολής είναι όπως στη C.

- Η εντολή ελέγχου `while (`c`)` s, όπου c πρέπει να είναι μια έγκυρη συνθήκη και s μια έγκυρη εντολή. Η σημασιολογία αυτής της εντολής είναι όπως στη C.

- Η εντολή άλματος `return` e `;` που τερματίζει την εκτέλεση της τρέχουσας συνάρτησης και επιστρέφει την τιμή της e ως αποτέλεσμα της συνάρτησης. Αν η τρέχουσα συνάρτηση έχει τύπο επιστροφής `proc` τότε η έκφραση e πρέπει να παραλείπεται. Διαφορετικά, η έκφραση e πρέπει να έχει τύπο ίδιο με τον τύπο επιστροφής της τρέχουσας συνάρτησης.

1.6 Βιβλιοθήκη έτοιμων συναρτήσεων

Η Alan υποστηρίζει ένα σύνολο προκαθορισμένων συναρτήσεων, οι οποίες έχουν υλοποιηθεί σε assembly του x86 ως μια *βιβλιοθήκη χρόνου εκτέλεσης* (run-time library). Είναι ορατές σε κάθε δομική μονάδα, εκτός αν επισκιάζονται από μεταβλητές, παραμέτρους ή άλλες συναρτήσεις με το ίδιο όνομα. Παρακάτω δίνονται οι τύποι τους και εξηγείται η λειτουργία τους.

1.6.1 Είσοδος και έξοδος

```
writeInteger (n : int)              : proc
writeByte    (b : byte)             : proc
writeChar    (b : byte)             : proc
writeString  (s : reference byte []) : proc
```

Οι συναρτήσεις αυτές χρησιμοποιούνται για την εκτύπωση τιμών που ανήκουν στους βασικούς τύπους της Alan, καθώς και για την εκτύπωση συμβολοσειρών. Οι `writeByte` και `writeChar` διαφέρουν ως προς το ότι η πρώτη εκτυπώνει την αριθμητική τιμή του byte ενώ η δεύτερη το χαρακτήρα που αντιστοιχεί σε αυτόν τον ASCII κωδικό.

```
readInteger ()                                : int
readByte    ()                                : byte
readChar    ()                                : byte
readString  (n : int, s : reference byte []) : proc
```

Αντίστοιχα, οι παραπάνω συναρτήσεις χρησιμοποιούνται για την εισαγωγή τιμών που ανήκουν στους βασικούς τύπους της Alan και για την εισαγωγή συμβολοσειρών. Η συνάρτηση `readString` χρησιμοποιείται για την ανάγνωση μιας συμβολοσειράς μέχρι τον επόμενο χαρακτήρα αλλαγής γραμμής. Οι παράμετροι της καθορίζουν το μέγιστο αριθμό χαρακτήρων (συμπεριλαμβανομένου του τελικού `'\0'`) που επιτρέπεται να διαβαστούν και τον πίνακα χαρακτήρων στον οποίο αυτοί θα τοποθετηθούν. Ο χαρακτήρας αλλαγής γραμμής δεν αποθηκεύεται. Αν το μέγεθος του πίνακα εξαντληθεί πριν συναντηθεί χαρακτήρας αλλαγής γραμμής, η ανάγνωση θα συνεχιστεί αργότερα από το σημείο όπου διακόπηκε.

1.6.2 Συναρτήσεις μετατροπής

```
extend (b : byte) : int
shrink (i : int)  : byte
```

Η πρώτη επεκτείνει μια τιμή τύπου `byte` στον αντίστοιχο αριθμό τύπου `int`. Η δεύτερη επιστρέφει μια τιμή τύπου `byte` που περιέχει τα 8 λιγότερο σημαντικά bits της παραμέτρου της.

1.6.3 Συναρτήσεις διαχείρισης συμβολοσειρών

```
strlen (s : reference byte [])                          : int
strcmp (s1 : reference byte [], s2 : reference byte [])  : int
strcpy (trg : reference byte [], src : reference byte []) : proc
strcat (trg : reference byte [], src : reference byte []) : proc
```

Οι συναρτήσεις αυτές έχουν ακριβώς την ίδια λειτουργία με τις συνώνυμές τους στη βιβλιοθήκη συναρτήσεων της γλώσσας C.

2 Πλήρης γραμματική της Alan

Η σύνταξη της γλώσσας Alan δίνεται παρακάτω σε μορφή EBNF. Η γραμματική που ακολουθεί είναι *διφορούμενη*, οι αμφισημίες όμως μπορούν να ξεπερασθούν αν λάβει κανείς υπόψη τους κανόνες προτεραιότητας και προσεταιριστικότητας των τελεστών, όπως περιγράφονται στον Πίνακα 2. Τα σύμβολα ⟨id⟩, ⟨int-const⟩, ⟨char-const⟩ και ⟨string-literal⟩ είναι τερματικά σύμβολα της γραμματικής.

⟨program⟩ ::= ⟨func-def⟩
⟨func-def⟩ ::= ⟨id⟩ “**(**” [⟨fpar-list⟩] “**)**” “**:**” ⟨r-type⟩ (⟨local-def⟩)* ⟨compound-stmt⟩
⟨fpar-list⟩ ::= ⟨fpar-def⟩ (“**,**” ⟨fpar-def⟩)*
⟨fpar-def⟩ ::= ⟨id⟩ “**:**” [“**reference**”] ⟨type⟩
⟨data-type⟩ ::= “**int**” | “**byte**”
⟨type⟩ ::= ⟨data-type⟩ [“**[**” “**]**”]
⟨r-type⟩ ::= ⟨data-type⟩ | “**proc**”
⟨local-def⟩ ::= ⟨func-def⟩ | ⟨var-def⟩
⟨var-def⟩ ::= ⟨id⟩ “**:**” ⟨data-type⟩ [“**[**” ⟨int-const⟩ “**]**”] “**;**”
⟨stmt⟩ ::= “**;**” | ⟨l-value⟩ “**=**” ⟨expr⟩ “**;**” | ⟨compound-stmt⟩ | ⟨func-call⟩ “**;**”
| “**if**” “**(**” ⟨cond⟩ “**)**” ⟨stmt⟩ [“**else**” ⟨stmt⟩]
| “**while**” “**(**” ⟨cond⟩ “**)**” ⟨stmt⟩ | “**return**” [⟨expr⟩] “**;**”
⟨compound-stmt⟩ ::= “**{**” (⟨stmt⟩)* “**}**”
⟨func-call⟩ ::= ⟨id⟩ “**(**” [⟨expr-list⟩] “**)**”
⟨expr-list⟩ ::= ⟨expr⟩ (“**,**” ⟨expr⟩)*
⟨expr⟩ ::= ⟨int-const⟩ | ⟨char-const⟩ | ⟨l-value⟩ | “**(**” ⟨expr⟩ “**)**” | ⟨func-call⟩
| (“**+**” | “**−**”) ⟨expr⟩ | ⟨expr⟩ (“**+**” | “**−**” | “*****” | “**/**” | “**%**”) ⟨expr⟩
⟨l-value⟩ ::= ⟨id⟩ [“**[**” ⟨expr⟩ “**]**”] | ⟨string-literal⟩
⟨cond⟩ ::= “**true**” | “**false**” | “**(**” ⟨cond⟩ “**)**” | “**!**” ⟨cond⟩
| ⟨expr⟩ (“**==**” | “**!=**” | “**<**” | “**>**” | “**<=**” | “**>=**”) ⟨expr⟩
| ⟨cond⟩ (“**&**” | “**|**”) ⟨cond⟩

3 Παραδείγματα

Στην παράγραφο αυτή δίνονται πέντε παραδείγματα προγραμμάτων στη γλώσσα Alan, η πολυπλοκότητα των οποίων κυμαίνεται σημαντικά. Για κάποια από αυτά τα παραδείγματα (ή για παρόμοια προγράμματα), μπορείτε να βρείτε τον αρχικό, τον ενδιάμεσο κώδικα (χωρίς βελτιστοποίηση), τη μορφή των εγγραφημάτων δραστηριοποίησης των δομικών μονάδων, καθώς και τον τελικό κώδικα σε αντίστοιχα φυλλάδια περιγραφής γλωσσών που δόθηκαν ως θέματα εργασίας στο ίδιο μάθημα σε προηγούμενα έτη, μέσω της ιστοσελίδας του μαθήματος.

3.1 Πες γεια!

Το παρακάτω παράδειγμα είναι ένα από τα απλούστερα προγράμματα στη γλώσσα Alan που παράγει κάποιο αποτέλεσμα ορατό στο χρήστη. Το πρόγραμμα αυτό τυπώνει απλώς ένα μήνυμα.

```
1   hello () : proc
2   {
3      writeString(”Hello world!\n”);
4   }
```

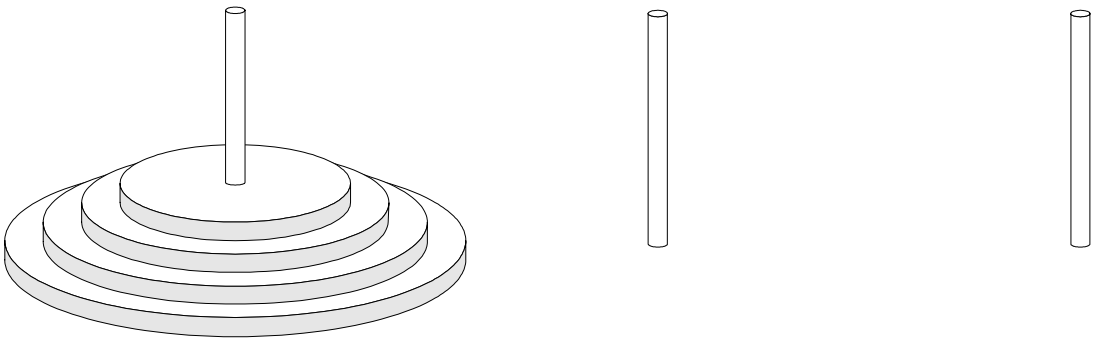
3.2 Οι πύργοι του Hanoi

Το πρόγραμμα που ακολουθεί λύνει το πρόβλημα των πύργων του Hanoi. Μια σύντομη περιγραφή του προβλήματος δίνεται παρακάτω.

Υπάρχουν τρεις στύλοι, στον πρώτο από τους οποίους είναι περασμένοι n το πλήθος δακτύλιοι. Οι εξωτερικές διάμετροι των δακτυλίων είναι διαφορετικές και αυτοί είναι περασμένοι από κάτω προς τα πάνω σε φθίνουσα σειρά εξωτερικής διαμέτρου, όπως φαίνεται στο Σχήμα 1. Ζητείται να μεταφερθούν οι δακτύλιοι από τον πρώτο στον τρίτο στύλο (χρησιμοποιώντας το δεύτερο ως βοηθητικό χώρο), ακολουθώντας όμως τους εξής κανόνες:

- Κάθε φορά επιτρέπεται να μεταφερθεί ένας μόνο δακτύλιος, από κάποιο στύλο σε κάποιον άλλο στύλο.
- Απαγορεύεται να τοποθετηθεί δακτύλιος με μεγαλύτερη διάμετρο πάνω από δακτύλιο με μικρότερη διάμετρο.

Το πρόγραμμα στη γλώσσα Alan που λύνει αυτό το πρόβλημα δίνεται στην αρχή της επόμενης σελίδας. Η συνάρτηση `hanoi` είναι αναδρομική.



Σχήμα 1: Οι πύργοι του Hanoi.

```
solve () : proc

   hanoi (rings : int, source : reference byte [],
          target : reference byte [], auxiliary : reference byte []) : proc

      move (source : reference byte [], target : reference byte[]) : proc
      {
         writeString("Moving from ");
         writeString(source);
         writeString(" to ");
         writeString(target);
         writeString(".\n");
      }

   { -- hanoi
      if (rings >= 1) {
         hanoi(rings-1, source, auxiliary, target);
         move(source, target);
         hanoi(rings-1, auxiliary, target, source);
      }
   } -- hanoi

   NumberOfRings : int;

{ -- solve
   writeString("Rings: ");
   NumberOfRings = readInteger();
   hanoi(NumberOfRings, "left", "right", "middle");
} -- solve
```

3.3 Πρώτοι αριθμοί

Το παρακάτω παράδειγμα προγράμματος στη γλώσσα Alan είναι ένα πρόγραμμα που υπολογίζει τους πρώτους αριθμούς μεταξύ 1 και n, όπου το n καθορίζεται από το χρήστη. Το πρόγραμμα αυτό χρησιμοποιεί έναν απλό αλγόριθμο για τον υπολογισμό των πρώτων αριθμών. Μια διατύπωση αυτού του αλγορίθμου σε ψευδογλώσσα δίνεται παρακάτω. Λαμβάνεται υπόψη ότι οι αριθμοί 2 και 3 είναι πρώτοι, και στη συνέχεια εξετάζονται μόνο οι αριθμοί της μορφής $6k \pm 1$, όπου k φυσικός αριθμός.

Κύριο πρόγραμμα

τύπωσε τους αριθμούς 2 και 3
για $t := 6$ μέχρι n με βήμα 6 κάνε τα εξής:
    αν ο αριθμός $t - 1$ είναι πρώτος τότε τύπωσέ τον
    αν ο αριθμός $t + 1$ είναι πρώτος τότε τύπωσέ τον

Αλγόριθμος ελέγχου (είναι ο αριθμός t πρώτος;)

αν $t < 0$ τότε έλεγξε τον αριθμό $-t$
αν $t < 2$ τότε ο t δεν είναι πρώτος
αν $t = 2$ τότε ο t είναι πρώτος
αν ο t διαιρείται με το 2 τότε ο t δεν είναι πρώτος
για $i := 3$ μέχρι $t/2$ με βήμα 2 κάνε τα εξής:
    αν ο t διαιρείται με τον i τότε ο t δεν είναι πρώτος
ο t είναι πρώτος

Το αντίστοιχο πρόγραμμα στη γλώσσα Alan βρίσκεται στην επόμενη σελίδα.

```
main () : proc

   prime (n : int) : int
      i : int;
   {
      if (n < 0)           return prime(-n);
      else if (n < 2)      return 0;
      else if (n == 2)     return 1;
      else if (n % 2 == 0) return 0;
      else {
         i = 3;
         while (i <= n / 2) {
            if (n % i == 0)
               return 0;
            i = i + 2;
         }
         return 1;
      }
   }

   limit   : int;
   number  : int;
   counter : int;

{ -- main
   writeString("Limit: ");
   limit = readInteger();
   writeString("Primes:\n");
   counter = 0;
   if (limit >= 2) {
      counter = counter + 1;
      writeInteger(2);
      writeString("\n");
   }
   if (limit >= 3) {
      counter = counter + 1;
      writeInteger(3);
      writeString("\n");
   }
   number = 6;
   while (number <= limit) {
      if (prime(number - 1) == 1) {
         counter = counter + 1;
         writeInteger(number - 1);
         writeString("\n");
      }
      if (number != limit & prime(number + 1) == 1) {
         counter = counter + 1;
         writeInteger(number + 1);
         writeString("\n");
      }
      number = number + 6;
   }

   writeString("\nTotal: ");
   writeInteger(counter);
   writeString("\n");
} -- main
```

3.4 Αντιστροφή συμβολοσειράς

Το πρόγραμμα που ακολουθεί στη γλώσσα Alan εκτυπώνει το μήνυμα "Hello world!" αντιστρέφοντας τη δοθείσα συμβολοσειρά.

```
main () : proc

   r : byte [32];

   reverse (s : reference byte []) : proc
      i : int;
      l : int;
   {
      l = strlen(s);
      i = 0;
      while (i < l) {
         r[i] = s[l-i-1];
         i = i+1;
      }
      r[i] = '\0';
   }

{ -- main
   reverse("\n!dlrow olleH");
   writeString(r);
} -- main
```

3.5 Ταξινόμηση με τη μέθοδο της φυσαλίδας

Ο αλγόριθμος της φυσαλίδας (bubble sort) είναι ένας από τους πιο γνωστούς και απλούς αλγορίθμους ταξινόμησης. Το παρακάτω πρόγραμμα σε Alan τον χρησιμοποιεί για να ταξινομήσει έναν πίνακα ακεραίων αριθμών κατ' αύξουσα σειρά. Αν x είναι ο πίνακας που πρέπει να ταξινομηθεί και n είναι το μέγεθός του (θεωρούμε σύμφωνα με τη σύμβαση της Alan ότι τα στοιχεία του είναι τα $x[0], x[1], \ldots x[n-1]$), μια παραλλαγή του αλγορίθμου περιγράφεται με ψευδοκώδικα ως εξής:

Αλγόριθμος της φυσαλίδας (bubble sort)

επανάλαβε το εξής:
    για i από 0 ως $n-2$
        αν $x[i] > x[i+1]$
            αντίστρεψε τα $x[i]$ και $x[i+1]$
όσο μεταβάλλεται η σειρά των στοιχείων του x

Το αντίστοιχο πρόγραμμα σε γλώσσα Alan είναι το εξής:

```
main () : proc

   bsort (n : int, x : reference int []) : proc

      swap (x : reference int, y : reference int) : proc
         t : int;
      {
         t = x;
         x = y;
         y = t;
      }

      changed : byte;
```

```
14          i : int;
15
16       { -- bsort
17          changed = 'y';
18          while (changed == 'y') {
19             changed = 'n';
20             i = 0;
21             while (i < n-1) {
22                if (x[i] > x[i+1]) {
23                   swap(x[i], x[i+1]);
24                   changed = 'y';
25                }
26                i = i+1;
27             }
28          }
29       } -- bsort
30
31       writeArray (msg : reference byte [], n : int, x : reference int []) : proc
32          i : int;
33       {
34          writeString(msg);
35          i = 0;
36          while (i < n) {
37             if (i > 0) writeString(", ");
38             writeInteger(x[i]);
39             i = i+1;
40          }
41          writeString("\n");
42       }
43
44       seed : int;
45       x    : int [16];
46       i    : int;
47
48    { -- main
49       seed = 65;
50       i = 0;
51       while (i < 16) {
52          seed = (seed * 137 + 220 + i) % 101;
53          x[i] = seed;
54          i = i+1;
55       }
56       writeArray("Initial array: ", 16, x);
57       bsort(16, x);
58       writeArray("Sorted array: ", 16, x);
59    } -- main
```

4 Οδηγίες για την παράδοση

Ο τελικός μεταγλωττιστής πρέπει να μπορεί να εξάγει κατά βούληση ενδιάμεσο και τελικό κώδικα. Εφόσον δεν έχουν καθορισθεί οι παράμετροι λειτουργίας `-f` ή `-i`, που εξηγούνται παρακάτω, ο μεταγλωττιστής θα δέχεται το πηγαίο πρόγραμμα από ένα αρχείο με οποιαδήποτε κατάληξη (πχ. `*.alan`) που θα δίνεται ως το μοναδικό του όρισμα. Ο ενδιάμεσος κώδικας θα τοποθετείται σε αρχείο με κατάληξη `*.imm` και ο τελικός σε αρχείο με κατάληξη `*.asm`. Τα αρχεία αυτά θα βρίσκονται στον ίδιο κατάλογο και θα έχουν το ίδιο κυρίως όνομα. Π.χ. από το πηγαίο αρχείο `/tmp/hello.alan` θα παράγονται τα `/tmp/hello.imm` και `/tmp/hello.asm`.

Το εκτελέσιμο του τελικού μεταγλωττιστή θα πρέπει να δέχεται τις παρακάτω παραμέτρους:

`-O` σημαία βελτιστοποίησης (προαιρετική).

`-f` πρόγραμμα στο standard input, έξοδος τελικού κώδικα στο standard output.

`-i` πρόγραμμα στο standard input, έξοδος ενδιάμεσου κώδικα στο standard output.

Επίσης, η τιμή που θα επιστρέφεται στο λειτουργικό σύστημα από το μεταγλωττιστή θα πρέπει να είναι μηδενική στην περίπτωση επιτυχούς μεταγλώττισης και μη μηδενική σε αντίθετη περίπτωση.

Για την εύκολη ανάπτυξη του μεταγλωττιστή προτείνεται η χρήση ενός αρχείου `Makefile` με τα εξής (τουλάχιστον) χαρακτηριστικά:

- Με απλό `make`, θα δημιουργεί το εκτελέσιμο του τελικού μεταγλωττιστή.
- Με `make clean`, θα σβήνει όλα ανεξαιρέτως τα αρχεία που παράγονται αυτόματα (π.χ. αυτά που παράγουν τα `bison` και `flex`, τα object files και το τελικό εκτελέσιμο).
- Με `make distclean`, θα σβήνει όλα τα παραπάνω και το τελικό εκτελέσιμο.

Ένα παράδειγμα ενός τέτοιου `Makefile`, υποθέτοντας ότι γλώσσα υλοποίησης είναι η C και γίνεται χρήση των εργαλείων `flex` και `bison` για την παραγωγή του μεταγλωττιστή της γλώσσας Alan, είναι το παρακάτω.

```
CC=gcc
CFLAGS=-Wall

alanc: lexer.o parser.o symbol.o general.o error.o symbol.o
	$(CC) $(CFLAGS) -o $@ $^ -lfl

lexer.c: lexer.l parser.h
	flex -s -o $@ $<

parser.c parser.h: parser.y
	bison -dv -o $@ $<

clean:
	$(RM) *.o parser.c parser.h lexer.c core *~

distclean: clean
	$(RM) alanc
```

Η μορφή εμφάνισης των τετράδων και του τελικού κώδικα θα πρέπει να είναι αυτή που προτείνεται στο βιβλίο και στις διαλέξεις. Επίσης να ληφθούν υπόψη οι παρακάτω οδηγίες:

- Ενδιάμεσος κώδικας: να υιοθετηθεί μορφοποίηση ισοδύναμη με

```
printf("%d: %s, %s, %s, %s\n", ...)
```

- Τελικός κώδικας: προσοχή να δοθεί στη στηλοθέτηση (indentation). Να ακολουθηθεί το παρακάτω υπόδειγμα:

```
ετικέττα: <tab> εντολή <tab> όρισμα-1, όρισμα-2
          <tab> εντολή <tab> όρισμα-1, όρισμα-2
```