MIXAL COMPILER

Ονοματεπώνυμο: Νικόλαος Ιλαρίδης
ΑΕΜ: 4524

September 20, 2024

Έχω υλοποιήσει έναν compiler που ακολουθεί τα παρακάτω στάδια: **λεξική** ανάλυση, **συντακτική** ανάλυση, **σημασιολογική** ανάλυση και τελικά **παράγει εκτελέσιμο** κώδικα MIXAL. Παρακάτω περιγράφω τα βασικά μέρη του και πώς λειτουργούν :

1 Λεξική Ανάλυση (lex.l)

Στη λεξική ανάλυση, χρησιμοποιώ το Flex για να διαβάζω τον πηγαίο κώδικα και να τον διαχωρίζω σε tokens, δηλαδή αναγνωριστικά, λέξεις-κλειδιά, αριθμούς και σύμβολα. Αυτή η διαδικασία μου επιτρέπει να "σπάσω" το πρόγραμμα σε μικρότερα μέρη που έχουν νόημα για την υπόλοιπη ανάλυση. Για κάθε token που αναγνωρίζεται, το περνάω στη συντακτική ανάλυση ώστε να συνεχιστεί η επεξεργασία του.

2 Συντακτική Ανάλυση (sydc1.y)

Στη συντακτική ανάλυση, χρησιμοποιώ το Bison για να ελέγχω αν τα tokens που παράχθηκαν από τη λεξική ανάλυση ακολουθούν τους κανόνες της γραμματικής που έχω ορίσει. Μέσα από αυτή τη διαδικασία, δημιουργώ το Δέντρο Σύνταξης (AST), το οποίο αναπαριστά τη δομή του προγράμματος σε δενδρική μορφή. Έτσι, μπορώ να κατανοήσω τη σειρά και τη σχέση των εντολών, των εκφράσεων και των δηλώσεων. Αυτό το AST είναι το βασικό εργαλείο που χρησιμοποιώ για να προχωρήσω στη σημασιολογική ανάλυση και στην παραγωγή κώδικα.

Η συντακτική ανάλυση που υλοποιώ βασίζεται σε συγκεκριμένους κανόνες γραμματικής, τους οποιους παραθέτω παρακάτω :

Η σύνταξη της «ΑΠΛΗΣ ΓΛΩΣΣΑΣ» ορίζεται από τη γραμματική χωρίς συμφραζόμενα:

```
1     program     ::= stmt_seq
2     stmt_seq    ::= stmt_seq ; stmt
3                   | stmt
4     stmt        ::= assign_stmt
5                   | if_stmt
6                   | repeat_stmt
7                   | read_stmt
8                   | write_stmt
9     assign_stmt ::= id := exp
10    read_stmt   ::= read exp
11    write_stmt  ::= write id
12    if_stmt     ::= if exp then stmt_seq end
13                  | if exp then stmt_seq else stmt_seq end
14    repeat_stmt ::= repeat stmt_seq until exp
15    factor      ::= ( exp )
16                  | number
17                  | id
18    term        ::= factor
19                  | term * factor
20                  | term / factor
21    simple_exp  ::= term
22                  | simple_exp + term
23                  | simple_exp - term
24    rel_exp     ::= simple_exp
25                  | rel_exp < simple_exp
26                  | rel_exp = simple_exp
27    exp         ::= rel_exp
```

3 Σημασιολογική Ανάλυση (symbol.c)

Στη σημασιολογική ανάλυση, ελέγχω το πρόγραμμα για τυχόν σημασιολογικά λάθη (semantic errors), όπως η χρήση μεταβλητών που δεν έχουν δηλωθεί ή οι τύποι δεδομένων που δεν ταιριάζουν σε μια πράξη. Παράλληλα, δημιουργώ και διαχειρίζομαι τον πίνακα συμβόλων, στον οποίο αποθηκεύω πληροφορίες για τις μεταβλητές, όπως τα ονόματα, τις τιμές τους, και τις θέσεις μνήμης που τους έχω αναθέσει. Ο πίνακας συμβόλων μου επιτρέπει να παρακολουθώ τις μεταβλητές και τις τιμές τους κατά τη διάρκεια της εκτέλεσης του προγράμματος.

4 Παραγωγή Κώδικα MIXAL (mixalgenerator.c)

Στην παραγωγή κώδικα MIXAL, έχω υλοποιήσει μια συνάρτηση για κάθε τύπο κόμβου (node) του Δέντρου Σύνταξης (AST). Κάθε συνάρτηση

παράγει τις αντίστοιχες εντολές MIXAL που αντιστοιχούν στις πράξεις και τις δηλώσεις που εκπροσωπεί ο κόμβος. Χρησιμοποιώ μια κεντρική συνάρτηση, την handleNode, η οποία διατρέχει το AST και καλεί την κατάλληλη συνάρτηση για κάθε τύπο κόμβου που συναντά. Αυτή η προσέγγιση μου επιτρέπει να μετατρέπω το σύνολο του προγράμματος σε κώδικα MIXAL, με βάση τις λογικές και αριθμητικές πράξεις, τις εντολές ελέγχου ροής, και τις δηλώσεις μεταβλητών που περιέχονται σε αυτό.

5 Εκτέλεση Compiler

Για να εκτελέσω το πρόγραμμα, χρησιμοποιώ το Makefile που έχω δημιουργήσει για τη διαχείριση της διαδικασίας. Με την εντολή make clean, διαγράφονται όλα τα αρχεία MIXAL και τα ενδιάμεσα αρχεία που έχουν παραχθεί από προηγούμενες εκτελέσεις του compiler, εξασφαλίζοντας ένα καθαρό περιβάλλον για την επόμενη εκτέλεση.

Στη συνέχεια, με την εντολή make test, εκτελείται αυτόματα ο lexer και ο parser, και δημιουργείται ο τελικός compiler με το GCC. Αφού ολοκληρωθεί η μεταγλώττιση, ο compiler δοκιμάζεται χρησιμοποιώντας ένα αρχείο εισόδου (input file) για να βεβαιωθώ ότι λειτουργεί σωστά.

Επιπλέον, στον φάκελο samples έχω περιλάβει κάποια test cases που καλύπτουν τις βασικές λειτουργίες του compiler, όπως η ανάλυση εκφράσεων, οι βρόχοι και οι δηλώσεις μεταβλητών. Στον φάκελο results, αποθηκεύω τα αποτελέσματα από τα πρώτα τρία input files, τα οποία αποτελούν δείγματα των παραγόμενων εξόδων του compiler.