Εργασία MIXAL COMPILER

Γλώσσες Προγραμματισμού & Μεταγλωττιστές

Δεληγιαννάκης Χαράλαμπος

https://github.com/iHaz32/MIXAL-Compiler

Τμήμα Πληροφορικής ΑΠΘ

Contents

1 Εισαγωγή

Η εργασία αποτελεί μια ολοκληρωμένη προσέγγιση στην ανάπτυξη ενός μεταγλωττιστή για μια υποθετική γλώσσα προγραμματισμού. Η διαδικασία περιλαμβάνει τέσσερις βασικές φάσεις: τη λεξική ανάλυση, τη συντακτική ανάλυση, τη σημασιολογική ανάλυση και την παραγωγή κώδικα assembler (MIXAL).

Στην πρώτη φάση, η λεξική ανάλυση αναγνωρίζει τα λεξικά στοιχεία του προγράμματος, όπως μεταβλητές, αριθμούς και χειριστές, και τα μετατρέπει σε μια ακολουθία token. Στη συνέχεια, η συντακτική ανάλυση εξετάζει τη δομή των προγραμμάτων και επιβεβαιώνει ότι ακολουθούν τους κανόνες της γλώσσας, δημιουργώντας ένα συντακτικό δέντρο. Ακολουθεί η σημασιολογική ανάλυση, η οποία ελέγχει την ορθότητα των εκφράσεων και των μεταβλητών, διασφαλίζοντας ότι όλες οι δηλώσεις και οι χρήσεις τους είναι έγκυρες. Τέλος, στην τέταρτη φάση, ο κώδικας μετατρέπεται σε MIXAL, τη γλώσσα μηχανής του υπολογιστή MIX, παρέχοντας τη δυνατότητα εκτέλεσης του προγράμματος.

Η εργασία προσφέρει μια πρακτική εφαρμογή των θεωρητικών εννοιών που σχετίζονται με τις γλώσσες προγραμματισμού και των μεταγλωττιστών.

2 Λεκτική Ανάλυση

Η λεκτική ανάλυση γίνεται στο αρχείο **lexc.l**. Πρόκειται για το πρώτο στάδιο της διαδικασίας μεταγλώττισης και αναγνωρίζει τα tokens. Αυτά μπορεί να είναι είτε λέξεις κλειδία της γλώσσας (πχ. if), είτε IDs (ονόματα μεταβλητών), είτε αριθμητικές ακέραιες σταθερές (DEC_CONST) είτε τελεστές και δίαφορα σύμβολα σύνταξης (πχ. +, ;) Λειτουργεί μεταξύ του πηγαίου κώδικα και των συμβόλων που χρησιμοποιούνται αργότερα από τον συνακτικό αναλυτή. Στο παραπάνω αρχείο, περιλαμβάνονται ο κανόνες της λεκτικής ανάλυσης, οι οποίοι καθορίζουν ποια τμήματα του κώδικα πρέπει να αναγνωριστούν ως συγκεκριμένα tokens.

Στο κύριο μέρος του αρχείου, ορίζονται οι κανόνες λεκτικής ανάλυσης χρησιμοποιώντας κανονικές εκφράσεις για να την αντιστοίχηση των διαφόρων tokens με το κείμενο εισόδου. Οι κανονικές εκφράσεις χρησιμοποιούνται για την ανίχνευση λέξεων-κλειδιών, αριθμητικών σταθερών, τελεστών και συμβόλων. Ένα ακόμα σημαντικό στοιχείο είναι η διαχείριση των κενών χαρακτήρων και σχολίων. Στην περίπτωση αυτή, το Lex αγνοεί χαρακτήρες όπως κενά, tabs, και νέες γραμμές.

Η διαδικασία της λεκτικής ανάλυσης ξεκινά όταν το αρχείο πηγαίου κώδικα περνάει από τον αναλυτή Lex. Η Lex δημιουργεί έναν αναγνωριστή (lexer), ο οποίος σαρώνει τον κώδικα και αναγνωρίζει τα tokens με βάση τους κανόνες που ορίστηκαν στο αρχείο lexc.l. Κάθε φορά που εντοπίζεται ένα token, ο αναγνωριστής επιστρέφει αυτό το token στον συντακτικό αναλυτή (bison parser).

3 Συντακτική Ανάλυση

Η συντακτική ανάλυση είναι το επόμενο στάδιο μετά τη λεκτική ανάλυση, όπου τα tokens που έχουν αναγνωριστεί από τον λεκτικό αναλυτή οργανώνονται σε μια δομή που αντιπροσωπεύει τη σύνταξη του προγράμματος, γνωστή ως συντακτικό δέντρο (syntax tree). Στο αρχείο **sydc.c** που ορίζει τον συντακτικό αναλυτή, χρησιμοποιείται το εργαλείο Bison για να κατασκευάσει αυτό το δέντρο, βασιζόμενο σε κανόνες (η γραμματική της γλώσσας δίνεται στην εκφώνηση της εργασίας) που καθορίζουν πώς τα tokens συνδυάζονται για να σχηματίσουν εντολές, εκφράσεις και δηλώσεις.

Οι κανόνες της συντακτικής ανάλυσης καθορίζουν πώς κάθε token (όπως αριθμοί, μεταβλητές, τελεστές) συνδυάζεται σε μεγαλύτερες δομές όπως εκφράσεις και εντολές ελέγχου ροής (π.χ., if-else, repeat-until). Το δέντρο που παράγεται αποτελείται από κόμβους (TreeNode), που μπορεί να αναπαριστούν ενέργειες (όπως ανάθεση τιμής) ή λογικές εκφράσεις.

Το συντακτικό δέντρο βοηθά στη διατήρηση της ιεραρχίας και της λογικής δομής του προγράμματος, καθώς κάθε κόμβος αντιστοιχεί σε μία πράξη του προγράμματος, και τα παιδιά του κόμβου αναπαριστούν τα επιμέρους μέρη αυτής της πράξης. Για παράδειγμα, μια πρόσθεση θα είχε ως κόμβο τον τελεστή + και ως παιδιά τους δύο τελεστέους.

Μετά την κατασκευή του συντακτικού δέντρου, η συνάρτηση expand_node επεξεργάζεται το δέντρο για να εκτελέσει το πρόγραμμα και να αξιολογήσει τις εκφράσεις, ενώ παράλληλα εκτελείται η σύνδεση με τον πίνακα συμβόλων για τη διαχείριση μεταβλητών.

4 Σημασιολογική Ανάλυση

Η σημασιολογική ανάλυση είναι το στάδιο που ακολουθεί τη συντακτική ανάλυση και επικεντρώνεται στη διασφάλιση ότι οι δομές και οι εντολές του προγράμματος έχουν νόημα βάσει των κανόνων της γλώσσας προγραμματισμού. Αυτό το στάδιο ασχολείται με τον έλεγχο τύπων δεδομένων, τη δήλωση και χρήση μεταβλητών, και την ορθή αντιστοίχιση μεταξύ των λειτουργιών και των παραμέτρων τους.

Στο αρχείο **symbol_table.c**, γίνεται η διαχείριση του πίνακα συμβόλων (symbol table), που είναι ένα κρίσιμο στοιχείο της σημασιολογικής ανάλυσης. Ο πίνακας συμβόλων καταγράφει όλες τις μεταβλητές που έχουν δηλωθεί, μαζί με πληροφορίες όπως το όνομα, την τιμή και τη θέση στη μνήμη. Μέσω της λειτουργίας create_symbol(), δημιουργούνται νέες εγγραφές για μεταβλητές, ενώ η λειτουργία find_symbol() επιτρέπει τον έλεγχο της ύπαρξης μιας μεταβλητής προτού χρησιμοποιηθεί, αποτρέποντας έτσι λάθη όπως η χρήση μη δηλωμένων μεταβλητών.

Κατά τη σημασιολογική ανάλυση, επιβεβαιώνεται ότι οι μεταβλητές έχουν δηλωθεί πριν τη χρήση τους και ότι οι πράξεις που πραγματοποιούνται (όπως αριθμητικές εκφράσεις ή αναθέσεις τιμών) είναι τύπου-συμβατές. Αυτό διασφαλίζεται με τη χρήση συναρτήσεων όπως η decide_expression() στο αρχείο syntax_tree.c, που αξιολογεί τις εκφράσεις και επιστρέφει τις τιμές τους βασισμένες στον πίνακα συμβόλων.

Συνοπτικά, η σημασιολογική ανάλυση εξασφαλίζει ότι το πρόγραμμα δεν παραβιάζει κανόνες όπως η χρήση μη δηλωμένων ή ασύμβατων τύπων μεταβλητών, και ότι έχει νόημα πέρα από τη σωστή σύνταξη, βοηθώντας στην αποτροπή λογικών λαθών στο πρόγραμμα.

5 Παραγωγή Κώδικα MIXAL

Η παραγωγή κώδικα MIXAL είναι η διαδικασία μετατροπής της αναλυμένης συντακτικής δομής του προγράμματος σε κώδικα που μπορεί να εκτελείται από τον υπολογιστή. Η MIXAL είναι ψευδογλώσσα που προσομοιώνει τις εντολές της αρχιτεκτονικής MIX, επιτρέποντας τη δημιουργία προγραμμάτων που εκτελούνται σε αυτό το εικονικό μηχάνημα.

Ο κώδικας που περιλαμβάνεται στο αρχείο **mix_codegen.c()** είναι υπεύθυνος για τη δημιουργία MIXAL κώδικα με βάση τη συντακτική δομή του προγράμματος που έχει αναλυθεί. Στην αρχή, η συνάρτηση createMixFile ανοίγει ένα αρχείο για την αποθήκευση του παραγόμενου κώδικα, ενώ η generate_mix_code() είναι η κύρια συνάρτηση που διαχειρίζεται την παραγωγή κώδικα για τους διάφορους τύπους κόμβων του δέντρου συντακτικής ανάλυσης.

Κάθε τύπος κόμβου (π.χ. εκφράσεις, δηλώσεις, συνθήκες) έχει τη δική του λογική παραγωγής κώδικα. Για παράδειγμα, οι αριθμητικές εκφράσεις, όπως οι προσθέσεις ή οι αφαιρέσεις, διαχειρίζονται από τη συνάρτηση generate_expression(), η οποία αναλύει τους αριστερούς και δεξιούς τελεστές και παράγει τον κατάλληλο MIXAL κώδικα για την εκτέλεση των υπολογισμών. Οι συνθήκες όπως οι IF και ELSE διαχειρίζονται με τη χρήση ετικετών (labels) για την κατεύθυνση της ροής του προγράμματος, επιτρέποντας την εκτέλεση του κώδικα μόνο αν πληρούνται συγκεκριμένα κριτήρια.

Η παραγωγή κώδικα περιλαμβάνει επίσης τη διαχείριση των μεταβλητών μέσω του πίνακα συμβόλων, που εξασφαλίζει ότι οι σωστές διευθύνσεις μνήμης χρησιμοποιούνται κατά την ανάγνωση και την εγγραφή τιμών. Έτσι, οι δηλώσεις εισόδου και εξόδου εκτελούνται μέσω των κατάλληλων εντολών του MIXAL, εξασφαλίζοντας ότι οι μεταβλητές αποθηκεύονται και ανακτώνται σωστά.

Συνολικά, ο κώδικας παραγωγής MIXAL συνδυάζει τις αρχές της συντακτικής ανάλυσης και της σημασιολογικής ανάλυσης, μετατρέποντας τις δομές ενός προγράμματος σε λειτουργικό κώδικα που μπορεί να εκτελείται, εξασφαλίζοντας την ορθότητα και την αποδοτικότητα των εντολών.

6 Τρόπος Εκτέλεσης

Ανοίγουμε το τερματικό και πληκτρολογούμε τις εξής windows εντολές:

1. flex lexc.l
2. bison -d sydc.y
3. gcc -o myparser sydc.tab.c lex.yy.c symbol_table/symbol_table.c syntax_tree/syntax_tree.c mixal/mix_codegen.c zyywrap.c
4. type testings/ONOMA_TESTING_ΑΡΧΕΙΟΥ.txt | ./myparser

Μετά από αυτά, θα δημιουργηθεί το αρχείο log.txt που περιλαμβάνει το συντακτικό δέντρο και άλλα απαραίτητα πράγματα, καθώς και το αρχείο mixal.mix που περιλαμβάνει τον κώδικα MIXAL του testing αρχείου που έτρεξε.

Για να τρέξουμε και το αρχείο MIXAL, ανοίγουμε τον emulator της επιλογής μας, βάζουμε το αρχείο mixal και πατάμε το κουμπί Run.

(Για το project επιλέχτηκε ο εξής emulator: https://github.com/rbergen/MixEmul)