Ονοματεπώνυμα:

Γεωργιάδης Γεώργιος 3199

Κουτρούδης Ιωάννης 3258

Μεταφραστές Τελική Αναφορά

Αρχικά στην αρχή του κώδικα μας έχουμε δηλώσει τα token_types μας ώστε να μην χρειάζεται να τα μην χρειάζεται μέσα στον κώδικα να τα κάνουμε σαν strings.

Ανοίγουμε στη συνέχεια το αρχείο που επιθυμεί ο χρήστης να εκτελεστεί ο κώδικας και διαβάζουμε τον πρώτο χαρακτήρα που περιέχει.

Επίσης έχουμε δηλώσει κάποιες global μεταβλητές για τον ενδιάμεσο κώδικα , τον τελικό κώδικα και τον πίνακα συμβόλων.

Λεκτικός Αναλυτής:

Τον λεκτικο αναλυτή τον υλοποιούμε μέσα στον lexer

Αρχικά κάθε φορά που καλούμε τον λεκτικό αναλυτή αρχικοποιούμε στο κενό τον πίνακα tokens και το string words.

Επίσης όσον αφορά τα κενά(space) , τις αλλαγές γραμμών(\n) και τα tabs τα παραλείπουμε και σε κάθε αλλαγή γραμμής βάζουμε έναν μετρητή και τον αυξάνουμε κατά ένα κάθε φορά για να ξέρουμε σε ποια γραμμή βρισκόμαστε σε περίπτωση λάθους.

Στην συνέχεια ελέγχουμε αν ο πρώτος χαρακτήρας μιάς λέξης είναι γράμμα και άν είναι ελέγχουμε αν ο επόμενος είναι γράμμα ή αριθμός και τα προσθέτουμε σε ένα κενό αλφαριθμητικό κάθε φορά για να ελέγξουμε την κάθε μία λέξη.

Επόμενη δουλειά μας είναι να ελέγξουμε αν αυτή η λέξη ανήκει στις ειδικές λέξεις και αν ανήκει την προσθέτουμε σε έναν πίνακα από strings με όνομα tokens,στον οποίο στην πρώτη του θέση υπάρχει το token_type και στην δεύτερη υπάρχει η λέξη όπως την διαβάσαμε.

Στην συνέχεια κάνουμε έλεγχο για διάβασμα αριθμού και αφού διαβάσουμε όλα τα στοιχεία του τα βάζουμε επίσης στον πίνακα.

Στην συνέχεια κάνουμε την ίδια δουλειά απλα αυτήν την φορά ελέγχουμε για όλα τα σύμβολα της γλώσσας minimal.

Αν τυχόν δεν αναγνωρίσει κανένα από όλα τα πραπάνω σημαίνει ότι έχουμε ολοκληρώσει το πρόγραμμα μας επιτυχώς και τυπώνουμε το αλφαριθμητικό End Of Program.

Τέλος αποθηκεύουμε σε 2 global μεταβλητές token_types και token_value ώστε να έχουμε πρόσβαση ανα πάσα στιγμή στο συγκεκριμένο και επιστρέφουμε το token_type.

Επιπλέον να σας ενημερώσουμε ότι δεν καταφέραμε να υλοποιήσουμε τον κώδικα για τα σχόλια.

Συνακτικός Αναλυτής:

Τον συντακτικό αναλυτή τον υλοποιούμε μέσα στον parser

Σύμφωνα με την γραμματική της γλώσσας minimal έχουμε υλοποιήσει τις κατάλληλες συναρτήσεις.

Κάθε φορά που καλούμε τον λεκτικό αναλυτή απόθηκεύουμε την επιστρεφόμενη τιμή μέσα σε ένα string tokens.Άν αυτό το token αντιστοιχεί σε κάποια από τις ειδικές λέξεις που θέλουμε να δούμε προχωράμε αλλιώς τυπώνουμε ένα error ή επιστρεφουμε πίσω.

Ενδιάμεσος Κώδικας:

Τον ενδιάμεσο κώδικα τον υλοποιούμε μέσα στον parser

Αρχικά έχουμε υλοποιήσει μερικές βοηθητικές συναρτήσεις στην αρχή του κώδικα οι οποίες μας βοηθάνε στην υλοποίηση των τετράδων.Οι συναρτήσεις αυτές είναι οι εξής:

Nextquad, genquad, newTemp, makelist, merge, backpatch.

Nextquad:Σε αυτήν την συνάρτηση επιστρέφουμε τον μετρητή της επόμενης τετράδας,ο οποίος αντιπροσωπεύει τον αριθμό της κάθε τετράδας.

Genquad:Σε αυτήν την συνάρτηση δημιουργούμε την τετράδα και την προσθέτουμε σε έναν πίνακα quad και ταυτόχρονα αυξάνουμε κατά ένα τον μετρητή των τετράδων.

newTemp:Σε αυτήν την συνάρτηση επιστρέφουμε το αλφαριθμητικό το οποίο αντιστοιχεί στην επόμενη προσωρινή μεταβλητή.

Makelist:Στην συνάρτηση αυτή δημιουργούμε μια λίστα ετικετών τετράδων που περιέχει μόνο τον αριθμό της ετικέτας.

Merge:Στην συνάρτηση αυτή συνενώνουμε δύο λίστες και επιστρέφουμε την νέα λίστα που περιέχει τις υπομέρους.

Backpatch:Σε αυτήν την συνάρτηση συμπληρώνουμε την τελευταία θέση της τετράδας ελέγχοντας αν αυτή είναι κενή.

Κάθε φορά που ξεκινάμε ένα πρόγραμμα ή ξεκινάμε μια καινούρια συνάρτηση φτιάχνουμε μια τετράδα με το όνομα του προγράμματος ή της συνάρτησης και στο πρώτο όρισμα βάζουμε το Begin_block και αντίστοιχα όταν κλείνει το block βάζουμε μία τετράδα με το όνομα του προγράμματος ή της συνάρτησης και στο πρώτο όρισμα βάζουμε το End_block.Επίσης όταν τελειώνει το πρόγραμμα βάζουμε και μια τετράδα με πρώτο όρισμα το halt.

Επιπλέον κατά την διαδικασία αριθμητικών πράξεων φτιάχνουμε μία τετράδα που στο πρώτο όρισμα βάζουμε την πράξη στο δεύτερο και τρίτο τους συντελεστές της πράξης αυτής και στο τέταρτο εκεί που αποθηκεύεται το αποτέλεσμα.Στην περίπτωση που μια διαδικασία πράξεων δεν χωράει σε μια τετράδα μόνο αποθηκεύουμε το αποτέλεσμα της πράξης με προτεραιότητα σε μια προσωρινή μεταβλητή και μετά κάνουμε την ακόλουθη πράξη συμπεριλαμβάνοντας την προσωρινή μεταβλητή.

Επιπρόσθετα στις αναθέσεις φτιάχνουμε μια τετράδα που στο πρώτο της όρισμα βάζουμε το σύμβολο της ανάθεσης(:=) ,στο δεύτερο όρισμα βάζουμε την τιμή που που θέλουμε να αναθέσουμε και στο τέταρτο βάζουμε τον συντελεστή που θα ανατεθεί η τιμή.

Όσον αφορά τις συνθήκες if,else,while,forcase ελέγχουμε ποια θα είναι η συνθήκη φτιάχνοντας μια τετράδα στην οποία στο πρώτο όρισμα βάζουμε το σύμβολο συνθήκης ,στο δεύτερο και τρίτο βάζουμε τις μεταβλητές-αριθμούς αριστερά και δεξιά της συνθήκης και στο τέταρτο βάζουμε τον αριθμό της τετράδας η οποία μας δείχνει τον αριθμό της τετράδας στην οποία πρέπει να μεταβεί αν ισχύει η συνθήκη.Επίσης κάτω από καθεμία από αυτές τις τετράδες προσθέτουμε μια επιπλέον τετράδα στην οποία το πρώτο όρισμα είναι το jump και στο τέταρτο έχουμε τον αριθμό της τετράδας που θα μεταβεί αν δεν ισχύει η παραπάνω συνθήκη.Τέλος στις συναρτήσεις while,forcase χρησιμοποιούμε την βοηθητική συνάρτηση Backpatch ώστε να διατρέξει τις ανάλογες τετράδες που δεν έχει συμπληρωθεί το τελευταίο όρισμα τους και το συμπληρώνει αναλόγως.

Όσον αφορά το return φτιάχνουμε μία τετράδα η οποία στο πρώτο όρισμα έχει το retv και στο δεύτερο περιέχει οτιδήποτε θέλουμε να επιστρέψουμε.

Για την συνάρτηση call φτιάχνουμε τις εξής τετράδες:

Μία τετράδα είναι για τα in/inout:

Για το in φτιάχνουμε μία τετράδα η οποία στο πρώτο της όρισμα έχει το par,στο δεύτερο έχει την μεταβλητή τύπου in και στο τρίτο έχει το CV.

Για το inout φτιάχνουμε μία τετράδα η οποία στο πρώτο της όρισμα έχει το par,στο δεύτερο έχει την μεταβλητή τύπου inout και στο τρίτο έχει το REF.

Τέλος φτιάχνουμε μια τετράδα που στο πρώτο της όρισμα περιέχει το call και στο δεύτερο το όνομα της συνάρτησης που θέλουμε να καλέσουμε.

Σε περίπτωση που θέλουμε να αποθηκεύσουμε το return που κάνει μια συνάρτηση σε μία μεταβλητή βάζουμε επιπλέον μια τετράδα στην οποία στο πρώτο όρισμα έχουμε το par ,στο δεύτερο έχουμε το όνομα της μεταβλητής που θέλουμε να αποθηκευτεί και στο τρίτο έχουμε το RET.

Όσον αφορά το print φτιάχνουμε μια τετράδα η οποία περιέχει στο πρώτο της όρισμα το out και στο δεύτερο το όνομα της μεταβλητής που θέλουμε να κάνουμε print.

Όσον αφορά το input φτιάχνουμε μια τετράδα η οποία περιέχει στο πρώτο της όρισμα το inp και στο δεύτερο το όνομα της μεταβλητής που που θα χρησιμοποιήσουμε σαν input.

Πίνακας Συμβόλων:

Τον πίνακα συμβόλων τον υλοποιούμε μέσα στον parser

Πρώτα από όλα δημιουρούμε έναν μετρητή για να ξέρουμε σε ποιο επίπεδο βρισκόμαστε π.χ. αν ξεκινήσει ένα πρόγραμμα βρισκόμαστε στο επίπεδο 0 , εάν ανοίξει μια συνάρτηση μέσα στο πρόγραμμα μεταβαίνουμε στο επίπεδο 1,εάν κλείσει η συνάρτηση καναγυρνάμε στο επίπεδο 0.

Επίσης φτιάχνουμε έναν πίνακα τον οποίο ονομάζουμε πίνακα συμβόλων τον οποίο χρησιμοποιούμε για να βάζουμε μέσα τα δεδομένα του πίνακα συμβόλων.Τον πίνακα αυτόν τον χωρίζουμε σε επίπεδα σύμφωνα με το παράδειγμα που δώσαμε βάζοντας ουσιαστικά πίνακες μέσα σε πίνακες.

Επίσης χρησιμοποιούμε και έναν ακόμη μετρητή τον οποίο αρχικοποιούμε στο 12 και είναι το offset μας και κάθε φορά που βάζουμε ένα επιπλέον στοιχείο στον πίνακα συμβόλων τον αυξάνουμε κατά 4.

Τέλος έχουμε έναν πίνακα offsetArray στον οποίο αποθηκεύουμε τα offset σε εμφωλευμένες συναρτήσεις ώστε να παίρνουμε το σωστο offset για καθεμία από αυτές.

Τελικός Κώδικας:

Έχοντας ήδη έτοιμες τις τετράδες μας και τον πίνακα συμβόλων υλοποιούμε μέσα στην κλάση assembly1 τον τελικό κώδικα.

Αρχικά μερικές global μεταβλητές οι οποίες είναι οι εξής:

Assembly:Είναι ένας πίνακας στον οποίο προσθέτουμε τις κατάλληλες εντολές Assembly χωρίζοντας τα σε επίπεδα.

flag1:Το χρησιμοποιούμε για να τυπώσουμε τις κατάληλλες εντολές Assembly ανάλογα με το αν ξεκινάει πρόγραμμα ή συνάρτηση.

counterEp:Το χρησιμοποιούμε ώστε να διατρέχουμε το κατάλληλο επίπεδο του πίνακα συμβόλων.

counterAssembly:Το χρησιμοποιούμε για να χωρίσουμε τον πίνακα Assembly σε ετικέτες(labels)

Επίσης για την διευκόλυνση μας υλοποιήσαμε 2 βοηθητικές συναρτήσεις.Η μία είναι η countAssembly και η άλλη η findOffset.

Στην countAssembly αυξάνουμε τον μετρητή counterAssembly κατά 1 και στην συνέχεια εισάγουμε μέσα στον πίνακα Assembly έναν ακόμα πίνακα που στην πρώτη θέση βάζουμε το όνομα του label και στην συνέχεια θα προσθέσουμε τα δεδομένα που θα μπούν στο αντίστοιχο label.

Στην findOffset βρίσκουμε το offset της μεταβλητής που έχουμε θέσει ώς όρισμα.Σε περίπτωση που αυτό δεν υπάρχει τυπώνει ένα error και τερματίζει το πρόγραμμα.

Ξεκινάμε διατρέχοντας τις τετράδες μια-μια και παράλληλα διατρέχουμε και τον πίνακα συμβόλων χρησιμοποιώντας τις global μεταβλητές τις οποίες προσαυξάνουμε.Επίσης κάθε φορά που

αναγνωρίζουμε μια καινούρια τετράδα καλούμε το countAssembly ώστε να προσθέσουμε το label της κάθε τετράδας και να κάνουμε την δουλειά που χρειάζεται κάθε τετράδα μέσα σε αυτό το label.

Σύμφωνα με τα παραδείγματα του μαθήματος για την κάθε μια τετράδα έχουμε προσθέσει και τις κατάλληλες εντολές Assembly.

Αν αναγνωρίσουμε το begin_block στην πρώτη θέση της τετράδας αυξάνουμε το flag1 κατά 1 ώστε να τυπώσουμε διαφορετικές εντολές Assembly ανάλογα με το αν βρισκόμαστε στην αρχή του προγράμματος ή στην αρχή μίας συνάρτησης.Επίσης αυξάνουμε τον counterEp ώστε να ξέρουμε σε ποιο επίπεδο στον πίνακα συμβόλων βρισκόμαστε.

Αν αναγνωρίσουμε το end_block στην πρώτη θέση της τετράδας κάνουμε τις ίδιες δουλειές με την begin_block με την διαφορά ότι τώρα μειώνουμε κατά 1 το flag1 και το counterEp.

Όσον αφορά τις εντολές των αριθμητικών πράξεων ελέγχουμε αν είναι αριθμός ή μεταβλητή το δεύτερο και τρίτο όρισμα της τετράδας.Αν είναι αριθμός προσθέτουμε σε έναν καταχωρητή προσωρινών τιμών τον αριθμό ως έχει με την εντολή li και αν μεταβλητή προσθέτουμε σε έναν καταχωρητή προσωρινών τιμών το offset της εντολής αυτής με την εντολή lw και στην συνέχεια ακολουθούμε τα παραδείγματα του μαθήματος ως έχουν.

Στην συνέχεια για τα σύμβολα σχέσεων χρησιμοποιούμε τα εξής:

= (beq)

< (blt)

> (bgt)

<= (ble)

>= (bge)

<> (bne)

Και κάνουμε τις ανάλογες πράξεις σε Assembly.

Τέλος για όλα τα υπόλοιπα κάνουμε τις ανάλογες εντολές Assembly ακολουθώντας πιστά τα παραδείγματα του μαθήματος.