Όνομα: Σιρινιάν Αράμ Εμμανουήλ

Ο στόχος αυτού του project είναι η κατασκευή ενός μεταφραστή με την γλώσσα Python. Το πρόγραμμα παίρνει σαν είσοδο ένα αρχείο <test>.ci, ένα πρόγραμμα της γλώσσας Ciscal και επιστρέφει ένα πρόγραμμα γλώσσας μηχανής. Το πρόγραμμα αυτό είναι δομημένο πάνω σε ορισμένες φάσεις που είναι απαραίτητες για να κατασκευαστεί σωστά ο τελικός κώδικας. Οι φάσεις αυτές είναι οι εξής: 1) Λεκτική Ανάλυση, 2) Συντακτική Ανάλυση, 3) Παραγωγή Ενδιάμεσου Κώδικα, 4) Πίνακας Συμβόλων, 5) Παραγωγή Τελικού Κώδικα.

Λεκτική Ανάλυση

Ο λεκτικός αναλυτής παίρνει σαν είσοδο το αρχικό πρόγραμμα χαρακτήρα-χαρακτήρα και επιστρέφει λεκτικές μονάδες. Το αρχείο LexicalAnalyzer.py είναι υπεύθυνο για την λεκτική ανάλυση.

LexicalAnalyzer.py:

Αρχικά αυτό που κάνει το πρόγραμμα είναι να ελέγχει τον αριθμό των παραμέτρων που παίρνει ο μεταφραστής δηλαδή ελέγχει αν του περνάμε το αρχείο με τον αρχικό κώδικα. Ύστερα το πρόγραμμα αρχικοποιεί όλες τις σταθερές που είναι απαραίτητες για την λειτουργεία του, μερικές από αυτές είναι: (myTokenMap το οποίο είναι μια λίστα που περιλαμβάνει όλες τις λεκτικές μονάδες της γλώσσας Ciscal μαζί με έναν αριθμό <id> το οποίο είναι μοναδικό για το καθένα), (identifiers το οποίο είναι μια λίστα που περιλαμβάνει όλες τις δεσμευμένες λέξεις της γλώσσας Ciscal), (currentFilePositionPath το οποίο είναι ένα string με την ονομασία του αρχείου όπου θα αποθηκεύετε ο αριθμός της στοίβας που διαβάστηκε ο τελευταίος χαρακτήρας του αρχικού προγράμματος), (currentFileLinePath το οποίο είναι ένα string με την ονομασία του αρχείου όπου θα αποθηκεύτε ο αριθμός της γραμμής που διαβάστηκε ο τελευταίος χαρακτήρας του αρχικού προγράμματος). Το αρχείο αυτό έχει μια μέθοδο που ονομάζετε lex() η οποία είναι και το κύριο μέρος αυτού του προγράμματος.

def lex():

Η συνάρτηση αυτή λειτουργεί σαν αυτόματο καταστάσεων υλοποιημένο με σειρά εντολών απόφασης το οποίο βρίσκετε μέσα σε ένα while loop. Η συνάρτηση lex() έχει μια τοπική μεταβλητή ακεραίων state. Όταν το state είναι ίσο με -1 τότε η επόμενη λεκτική μονάδα παράχθηκε επιτυχώς η while loop το ελέγχει και τερματίζεται. Όταν το state είναι ίσο με -2 τότε υπάρχει κάποιο λάθος πχ. μη επιτρεπτός χαρακτήρας, μη έγκαιρη ονομασία παραμέτρου ή λάθος κατασκευή σχολείων. Στη τοπική μεταβλητή strResult τύπου string αποθηκεύετε η λεκτική μονάδα. Στη τοπική μεταβλητή inputCharacter τύπου string αποθηκεύετε ο επόμενος χαρακτήρας του αρχικού προγράμματος. Στη τοπική μεταβλητή fileCounter τύπου ακεραίων αποθηκεύετε η στήλη στην οποία θα διαβαστεί ο επόμενος χαρακτήρας. Στη τοπική μεταβλητή lineCounter τύπου ακεραίων αποθηκεύετε η γραμμή στην οποία θα διαβαστεί ο επόμενος χαρακτήρας. Κάθε φορά που εκτελείτε η συνάρτηση η τιμές των fileCounter, lineCounter και inputCharacter ανανεώνονται από τα αρχεία στα οποία είναι αποθηκεμένα.

Συντακτική Ανάλυση

Είναι το κομμάτι του προγράμματος όπου γίνεται ο έλεγχος για να διαπιστωθεί εάν το πηγαίο πρόγραμμα ανήκει ή όχι στη γλώσσα. Ο συντακτικός αναλυτής δουλεύει με αναδρομική κατάβαση και βασίζεται σε γραμματική LL(1). Η γραμματική LL(1) αναγνωρίζει από αριστερά στα δεξιά, την αριστερότερη δυνατή παραγωγή και όταν βρίσκεται σε δίλλημα ποιόν κανόνα να ακολουθήσει της αρκεί να κοιτάξει το αμέσως επόμενο σύμβολο στην συμβολοσειρά.

SyntaxAnalyzer.py:

Για κάθε κανόνα της γραμματικής υπάρχει μια συνάρτηση, όταν συναντάει μη τερματικό σύμβολο καλείται η αντίστοιχη συνάρτηση (syntaxAnalyzer(), program(), block(), declarations(), varlist(), subprograms(), func(), funcbody(), formalpars(), formalparlist(), formalparitem(), sequence(), bracketsSeq(), brackOrStat(), statement(), assignmentStat(), ifStat(), elsepart(), whileStat(), selectStat(), doWhileStat(), exitStat(), returnStat(), printStat(), callStat(), actualpars(), actualparlist(), actualparitem(), condition(), boolterm(), boolfactor(), expression(), term(), factor(), idtail(), relationalOper(), addOper(), mulOper(), optionalSign()). Στο αρχείο αυτό υπάρχει επίσης η συνάρτηση updateToken() η οποία καλεί την lex() (δηλ. τον λεκτικό αναλυτή) και τον αποθηκεύει στην global μεταβλητή token. Όταν συναντάει τερματικό σύμβολο και ο λεκτικός αναλυτής επιστρέφει λεκτική μονάδα που αντιστοιχεί στο τερματικό αυτό σύμβολο έχει αναγνωρίσει επιτυχώς τη λεκτική μονάδα. Αντίθετα όταν συναντάει τερματικό σύμβολο και ο λεκτικός αναλυτής δεν επιστρέφει τη λεκτική μονάδα που περιμένει ο συντακτικός αναλυτής εκτυπώνετε μήνυμα λάθους.

Παραγωγή Ενδιάμεσου Κώδικα

Ο ενδιάμεσος κώδικας είναι ένα σύνολο από τετράδες. Οι τετράδες είναι αριθμημένες. Κάθε τετράδα έχει μπροστά της έναν μοναδικό αριθμό που τη χαρακτηρίζει. Μόλις τελειώσει η εκτέλεση μίας τετράδας εκτελείται η τετράδα που έχει τον αμέσως μεγαλύτερο αριθμό, εκτός εάν η τετράδα που μόλις εκτελέστηκε υποδείξει κάτι διαφορετικό. Ο ενδιάμεσος κώδικας αποθηκεύετε σε ένα αρχείο το ./intermediateCode.int.

IntermediateCodeProduction.py:

Στο αρχείο αυτό είναι αποθηκευμένες πολλές βοηθητικές υπορουτίνες για την παραγωγή του ενδιάμεσου κώδικα. Αρχικά γίνεται η δήλωση των global μεταβλητών. Η global μεταβλητή foursome είναι η λίστα στην οποία αποθηκεύεται προσωρινά η επόμενη τετράδα του ενδιάμεσου κώδικα. Η global μεταβλητή foursomeBuffer είναι η λίστα στην οποία αποθηκεύονται προσωρινά οι επόμενες τετράδες του ενδιάμεσου κώδικα μαζί με τον μοναδικό αριθμό που τους χαρακτηρίζει. Η global μεταβλητή quadNumber είναι ο μοναδικός ακέραιος αριθμός της προηγούμενης τετράδας. Η global μεταβλητή tempVariableNumber είναι ο ακέραιος αριθμός που χρησιμοποιήθηκε για την κατασκευή της προσωρινής μεταβλητής για την προηγούμενη τετράδα. Οι βοηθητικές υπορουτίνες αυτού του αρχείου χρησιμοποιούνται στο αρχείο ./SyntaxAnalyzer.py ταυτοχρόνως δηλαδή όταν γίνεται η συντακτική ανάλυση γίνεται και η παραγωγή του ενδιάμεσου κώδικα.

def nextquad():

Δημιουργεί και επιστρέφει τον αριθμό της επόμενης τετράδας που πρόκειται να παραχθεί

def genquad(op, x, y, z):

Δημιουργεί την επόμενη τετράδα (op, x, y, z).

def newtemp():

Δημιουργεί και επιστρέφει μία νέα προσωρινή μεταβλητή. Οι προσωρινές μεταβλητές είναι της μορφής T_1, T_2, T_3, …

def emptylist():

Δημιουργεί μία κενή λίστα ετικετών τετράδων.

def makelist(x):

Δημιουργεί μία λίστα ετικετών τετράδων που περιέχει μόνο το x.

def merge(list1, list2):

Δημιουργεί μία λίστα ετικετών τετράδων από τη συνένωση των λιστών list1, list2.

def clearIntermediateCodeFile():

Σβήνει τυχόν δεδομένα από το αρχείο ./intermediateCode.int.

def saveResults():

Αποθηκεύει την προσωρινή τετράδα foursome μαζί με τον αριθμό που την χαρακτηρίζει στο foursomeBuffer.

def backpatch(inList, z):

Η λίστα inList αποτελείται από δείκτες σε τετράδες των οποίων το τελευταίο τελούμενο δεν είναι συμπληρωμένο. Η backpatch επισκέπτεται μία μία τις τετράδες αυτές και τις συμπληρώνει με την ετικέτα z.

def writeIntermediateCode():

Γράφει όλες τις τετράδες που είναι αποθηκευμένες στο foursomeBuffer στο αρχείο ./intermediateCode.int.

IntToC.py:

Στο αρχείο αυτό υπάρχουν όλες οι απαραίτητες συναρτήσεις ώστε ο ενδιάμεσος κώδικας που είναι αποθηκευμένες στο αρχείο ./intermediateCode.int να μετατραπεί εντολή προς εντολή σε κώδικα της γλώσσας προγραμματισμού C στο αρχείο ./test.c. Η μετατροπή αυτή δεν γίνετε όταν ο αρχικός κώδικας έχει μέσα του συναρτήσεις.

def intermediateCodeToCCode():

Αυτή είναι η συνάρτηση που χρησιμοποιεί τις υπόλοιπες συνάρτησης του αρχείου αυτού για να κατασκευάσει τον ενδιάμεσο κώδικα στην C και να τον αποθηκεύσει στο αρχείο ./test.c.

def decodeAndWriteIntermediateCode():

Η συνάρτηση αυτή διαβάζει από τις global λίστες tempList1, tempList2 και tempList3 κομμάτια πληροφορίας του ενδιάμεσου κώδικα και χρησιμοποιώντας πολλά επίπεδα από if δημιουργεί κατάλληλα string αρχεία για να πάνε αμέσως μετά να γραφτούν στο αρχείο ./test.c.

def writeTags(index1):

Δημιουργεί και γράφει μια νέα ετικέτα για κάθε νέα εντολή. Η ετικέτες έχουν την μορφή L_1, L_2, L_3, …

def writeLastProgramLines():

Γραφεί ότι χρειάζεται το πρόγραμμα ./test.c στο τέλος του αρχείου για να μπορέσει να τρέξει σωστά.

def writeFirstProgramLines():

Γραφεί ότι χρειάζεται το πρόγραμμα ./test.c στην αρχή του αρχείου για να μπορέσει να τρέξει σωστά.

def writeVariableInitialization():

Ξαναγράφει στο αρχείο ./test.c ότι είχε γραμμένο προσθέτοντας στην σωστή γραμμή του κώδικα μια γραμμή όπου αρχικοποιούνται όλες οι μεταβλητές που χρειάζονται για να εκτελεστεί το πρόγραμμα χωρίς πρόβλημα.

def checkForNewVariable(stringThatMayBeInt):

Ελέγχει αν το stringThatMayBeInt είναι αριθμός και αν δεν είναι τότε το προσθέτει στην global λίστα variableBuffer.

Πίνακας Συμβόλων

Ο πίνακας συμβόλων είναι μία δομή η οποία αποθηκεύει όλες τις μεταβλητές του προγράμματος ώστε να μπορεί ο τελικός κώδικας να τις βρίσκει διαβάζοντας την κατάλληλη διεύθυνση τους. Ο πίνακας εμφανίζετε στο τερματικό μέσω εντολών εκτύπωσης.

SymbolArray.py:

Σε αυτό το αρχείο υπάρχουν χρήσιμες κλάσεις που υλοποιούν τον πίνακα συμβόλων (class Scope, class Entity, class Argument, class SymbolArray).

class Scope:

Προσθήκη νέου Scope γίνεται όταν ξεκινάμε τη μετάφραση μιας νέας συνάρτησης. Ένα Scope διαγράφετε όταν τελειώνει η μετάφραση μιας συνάρτησης, με τη διαγραφή διαγράφεται η εγγραφή (record) του Scope και όλες οι λίστες με τα Entity και τα Argument που εξαρτώνται από αυτήν. Έχει μια παράμετρο την nestingLevel που είναι το βάθος φωλιάσματος.

def toString(self):

Η συνάρτηση αυτή επιστρέφει ένα string με την τιμή του nestingLevel.

class Entity:

Προσθέτετε νέο Entity όταν γίνεται δήλωση μεταβλητής, όταν δημιουργείται νέα προσωρινή μεταβλητή, όταν γίνεται δήλωση νέας συνάρτησης, όταν γίνεται δήλωση τυπικής παραμέτρου συνάρτησης. Όταν δημιουργείτε η συνάρτηση παίρνει μία παράμετρο entityType η οποία είναι ένα string και καθορίζει τον τύπο του Entity (variable ή function ή procedure ή constant ή parameter ή temporary).

def setEntityTypeVariable(self, name):

Η συνάρτηση αυτή καλείτε όταν το Entity είναι τύπου variable. Παίρνει ως παράμετρο το όνομα της μεταβλητής και θέτει την μεταβλητή της κλάσεις name με αυτήν. Θέτει την τιμή της παραμέτρου offset με αυτήν της sp. Ανανεώνει την τιμή του sp προσθέτοντας τον με το 4.

def setEntityTypeFunction(self, name):

Η συνάρτηση αυτή καλείτε όταν το Entity είναι τύπου function. Παίρνει ως παράμετρο το όνομα της συνάρτησης και θέτει την μεταβλητή της κλάσεις name με αυτήν.

def setEntityTypeProcedure(self, name):

Η συνάρτηση αυτή καλείτε όταν το Entity είναι τύπου procedure. Παίρνει ως παράμετρο το όνομα της procedure και θέτει την μεταβλητή της κλάσεις name με αυτήν.

def setEntityTypeParameter(self, name, parMode):

Η συνάρτηση αυτή καλείτε όταν το Entity είναι τύπου parameter. Παίρνει ως παράμετρο το όνομα της μεταβλητής και θέτει την μεταβλητή της κλάσεις name με αυτήν. Παίρνει ως παράμετρο τον τύπο parMode και την θέτει στην τοπική παράμετρο parMode. Θέτει την

τιμή της παραμέτρου offset με αυτήν της sp. Ανανεώνει την τιμή του sp προσθέτοντας τον με το 4.

def setEntityTypeTemporaryVariable(self, name):

Η συνάρτηση αυτή καλείτε όταν το Entity είναι τύπου temporary. Παίρνει ως παράμετρο το όνομα της μεταβλητής και θέτει την μεταβλητή της κλάσεις name με αυτήν. Θέτει την τιμή της παραμέτρου offset με αυτήν της sp. Ανανεώνει την τιμή του sp προσθέτοντας τον με το 4.

def toString(self):

Η συνάρτηση αυτή επιστρέφει ένα string με τις τιμές των μεταβλητών της συνάρτησης Entity ελέγχοντας την τιμή της μεταβλητής entityType.

class Argument:

Προσθήκη νέου Argument όταν γίνεται δήλωση τυπικής παραμέτρου συνάρτησης.

class SymbolArray:

Είναι μια λίστα που δομή μέσα της τον πίνακα συμβόλων.

def createNewLayer(self):

Κατασκευάζει ένα αντικείμενο Scope το tempScope και μία λίστα newLayerList στην οποία και εισάγετε και η tempScope. Έπειτα η συνάρτηση εισάγει το newLayerList στην λίστα της κλάσης symbolist. Στο τέλος ανανεώνει την τιμή της global μεταβλητής sp με 12 και εκτελεί τη συνάρτηση toString() της κλάσεις SymbolArray.

def insertNewEntity(self, entity):

Η συνάρτηση αυτή εισάγει στην τελευταία λίστα της λίστας symbolList την παράμετρο entity και μετά εκτελεί την συνάρτηση toString() της κλάσης SymbolArray.

def deleteLastLayer(self):

Η συνάρτηση αυτή διαγράφει την τελευταία λίστα της λίστας symbolList.

def toString(self):

Η συνάρτηση αυτή εκτυπώνει στο terminal τον πίνακα συμβόλων.

Παραγωγή Τελικού Κώδικα

Το κομμάτι της παραγωγής τελικού κώδικα δεν έχει υλοποιημένο στο πρόγραμμα αυτό αν και η παρουσία του είναι αναγκαία για την ολοκλήρωση του μεταφραστή. Ο ενδιάμεσος κώδικας είναι η είσοδος για την παραγωγή τελικού κώδικα. Από κάθε μια εντολή ενδιάμεσου κώδικα παράγονται οι αντίστοιχες εντολές του τελικού κώδικα. Οι μεταβλητές απεικονίζονται στη μνήμη, το πέρασμα παραμέτρων και η κλήση συναρτήσεων. Ο κώδικας που δημιουργείται είναι για τον επεξεργαστή MIPS.