Μεταφραστής Cimple

Μεταφραστές ΜΥΥ802

ΚΩΝΣΤΑΝΤΙΝΟΣ ΚΙΚΙΔΗΣ (4387)
ΚΩΝΣΤΑΝΤΙΝΟΣ ΤΣΑΜΠΙΡΑΣ (4508)

Εαρινό εξάμηνο 2021-2022

Προγράμματα που περνάνε από τον compiler μας

countDigits.ci, factorial.ci, primes.ci, summation.ci, finalCodeExample.ci, test.ci, test1.ci, test2.ci

Global μεταβλητές που χρησιμοποιήσαμε

Για τον τελικό κώδικα	
final = []	Οι γραμμές assembly κώδικα, κάθε καταχώριση στον πίνακα είναι μία γραμμή assembly που θα γραφεί σε ένα .asm αρχείο στο τέλος.
paramFlag = False	Boolean μεταβλητή που υποδηλώνει αν έχει διαβαστεί παράμετρος αλλά δεν έχει γίνει κλήση της call ακόμα.
paramList = []	Λίστα που μετρά την σειρά και το πλήθος των παραμέτρων.
Για τον πίνακα συμβόλων	
depth = -1	Το βάθος του τρέχοντος Scope.
table = []	Τα Scope του προγράμματος.
symTable = []	Οι γραμμές του πίνακα συμβόλων.
ret = False	Boolean μεταβλητή που υποδηλώνει την ύπαρξη return σε ένα υποπρόγραμμα.
Για τον ενδιάμεσο κώδικα	
label = 0	Αριθμός ετικέτας.
quadList = []	Λίστα από Quads.
tempCounter = 0	Μετρητής προσωρινών μεταβλητών.
variables = []	Λίστα προσωρινών μεταβλητών.
main_name = ""	Το όνομα του προγράμματος (program **xyz**)
fname = ""	Το όνομα της συνάρτησης που μεταφράζεται εκείνη την στιγμή.
flag = False	Boolean μεταβλητή που δείχνει αν είναι function (με False) ή procedure (με True)
Για τον λεκτικό & συντακτικό αναλυτή	
line = 1	Μετρητής γραμμών του προγράμματος.
all_tokens = []	Πίνακας με όλα τα Token της λεκτικής ανάλυσης.
token = 0	Το τρέχον token, το στοιχείο που εξετάζεται την συγκεκριμένη στιγμή.
index = -1	Η θέση του τρέχοντος token μέσα στον πίνακα.
eof_flag = False	Boolean μεταβλητή για το αν τελείωσε η ανάγνωση του αρχείου.
keywords = {…}	Οι δεσμευμένες λέξεις της γλώσσας Cimple.

Κλάσεις που χρησιμοποιήσαμε

Class Token	Για την αποθήκευση πληροφοριών ενός Token, δηλαδή το Token που αναγνωρίστηκε (recognized_string), το είδος του Token (family) και την γραμμή που εντοπίστηκε (line_number).
Class Scope	Για την αποθήκευση πληροφορίων που αφορούν ένα Scope, δηλαδή τα entity που έχει το συγκεκριμένο Scope (entityList), το επίπεδο φωλιάσματος (nestingLevel) του Scope και το Offset του (offset).
Class Entity	Για την αποθήκευση πληροφορίων που αφορούν ένα Entity, δηλαδή το όνομα (name) και τον τύπο (type) του Entity, το quad (startingQuad) από το οποίο ξεκινάει το Entity, τα arguments (args) που μπορεί να έχει, το offset του (offset) και το είδος περάσματος της μεταβλητής (parameterMode, για όταν είναι τύπου in/inout).
Class Argument	Για την αποθήκευση των παραμέτρων που έχουμε σε ένα Scope (τα τοποθετούμε στο τελευταίο Entity), δηλαδή το είδος της παραμέτρου (parmode) και έναν δείκτη στο επόμενο Argument (nextArg).

Λειτουργία του μεταφραστή

Στην αρχή, αρχικοποιούνται οι καθολικές μεταβλητές μας και ανοίγουμε το αρχείο που θέλουμε να μεταφράσουμε. Στη συνέχεια καλούμε την συνάρτηση main(), η οποία για όσο δεν έχει βρει end of file, παράγει νέα token μέσω της συνάρτησης lex(), στη συνέχεια καλούμε τον συντακτικό αναλυτή (syntax_analyzer()) ο οποίος πραγματοποιεί συντακτική ανάλυση του κώδικα και δημιουργεί όπου και όταν χρειάζεται τον ενδιάμεσο κώδικα, τον πίνακα συμβόλων και τον τελικό κώδικα. Με την ολοκλήρωση της συντακτικής ανάλυσης, εφόσον δεν έχει προηγηθεί κάποιο σφάλμα, καλούνται με την σειρά οι intercodeGen(), cCodeGen(), writeSymTable() και writeRiscVAsm() για την εγγραφή του ενδιάμεσου κώδικα, του κώδικα C, του πίνακα συμβόλων και του τελικού κώδικα σε αρχεία .int, .c, .symb και .asm αντίστοιχα.

Λεκτικός Αναλυτής

Η λεκτική ανάλυση αποτελεί την πρώτη φάση της μεταγλώττισης. Κατά τη φάση αυτή, διαβάζεται το *αρχικό πρόγραμμα* (το οποίο συνηθίζεται να ονομάζεται και *πηγαίο πρόγραμμα*) και παράγονται οι λεκτικές μονάδες. Χρησιμοποιούμε τον όρο *λεκτική μονάδα* για να αναπαραστήσουμε οτιδήποτε έχει νόημα να θεωρηθεί ως αυτόνομο σύνολο συνεχόμενων χαρακτήρων που μπορεί να συναντηθεί σε ένα πρόγραμμα και βρίσκει σημασιολογία στην υπό υλοποίηση γλώσσα.

def lex()	Ελέγχει τους χαρακτήρες έναν έναν και αναλόγως τι θα αναγνωρίσει, θα επιστρέψει το string που αναγνώρισε, την οικογένεια του (τι είδους string είναι) και την γραμμή στην οποία εντοπίστηκε.
def comment(char1,char2,line)	Ελέγχει αν βρέθηκε ο χαρακτήρας «#», ο οποίος υποδηλώνει την έναρξη comment, σταματά μόλις βρει το 2^{ο} «#» που στην ουσία κλείνει το comment.

Συντακτικός Αναλυτής

Τη φάση της λεκτικής ανάλυσης ακολουθεί η φάση της συντακτικής ανάλυσης. Κατά τη συντακτική ανάλυση ελέγχεται εάν η ακολουθία των λεκτικών μονάδων που σχηματίζεται από τον λεκτικό αναλυτή, αποτελεί μία νόμιμη ακολουθία με βάση τη γραμματική της γλώσσας.

def syntax_analyzer()	Ξεκινά την συντακτική ανάλυση, διαβάζει το 1^{ο} token, και καλεί την program(), με την ολοκλήρωση της θα εμφανίζει το μήνυμα επιτυχούς μετάφρασης (εφόσον δεν υπάρχει κάποιο σφάλμα).
def get_token()	Θα επιστρέψει το επόμενο token.
def error(typeOfError)	Θα σταματήσει την εκτέλεση του συντακτικού αναλυτή λόγω κάποιου συντακτικού σφάλματος, θα τυπώσει το κατάλληλο μήνυμα και θα τερματίσει την εκτέλεση του προγράμματος.
def actualparitem()	Υπεύθυνη για την αναγνώριση των keywords “in” και “inout”, αν αναγνωρίσει “in”, θα διαβάσει το expression και θα επιστρέψει την αποτίμηση της συνάρτησης ως “cv” (με αντίγραφο), διαφορετικά, θα την επιστρέψει ως “ref” (με αναφορά).
def actualparlist()	Με την ολοκλήρωση ανάγνωσης των παραμέτρων in/inout, φτιάχνει μία προσωρινή λίστα με όλα τα in και inout της συνάρτησης που την κάλεσε και αναλαμβάνει την παραγωγή των αντίστοιχων genQuad(“par”,…).
def addoperator()	Διαβάζει τα σύμβολα “+” και “-“.

def assignStat()	Υπεύθυνο για την ανάθεση μίας τιμής σε μία μεταβλητή και την δημιουργία του κατάλληλου genQuad(":=",…).
def block()	Υπεύθυνο για την δημιουργία των genQuad(begin_block,…), genQuad(end_block,…), genQuad(halt,…) για το κύριο πρόγραμμα. Αναλαμβάνει τα declarations, subprograms και blockstatements και με την ολοκλήρωση της παράγει τον πίνακα συμβόλων εκείνης της στιγμής, την παραγωγή του τελικού κώδικα για την main και την διαγραφή του τελευταίου scope.
def blockfuncproc(id)	Υπεύθυνο για τα block των υποπρογραμμάτων, αντίστοιχη της block.
def blockstatements()	Καλεί την statement() και όσες άλλες statement() ακολουθούν.
def boolfactor()	Υπεύθυνο για την αναγνώριση λογικών παραγόντων και την κατάλληλη αποτίμηση τους σε λίστες ετικετών eCondTrue και eCondFalse, οι οποίες επιστρέφονται με την ολοκλήρωση της κλήσης.
def boolterm()	Διαβάζει τα "and" της λογικής πρότασης, κάνει backpatch στα κατάλληλα σημεία προκειμένου να συμπληρωθούν κατάλληλα οι quad και συμπληρώνει και επιστρέφει τις λίστες conditionTrue και conditionFalse.
def callStat()	Διαβάζει το "call" και καλεί την actualparlist(), για την αναγνώριση των παραμέτρων. Δημιουργεί τα κατάλληλα genQuad όπου χρειάζεται.
def condition()	Διαβάζει τα "or" της λογικής πρότασης, κάνει backpatch στα κατάλληλα σημεία προκειμένου να συμπληρωθούν κατάλληλα οι quad και συμπληρώνει και επιστρέφει τις λίστες conditionTrue και conditionFalse.
def declarations()	Καλεί την varlist() και διαβάζει τις δηλώσεις των μεταβλητών που θα χρησιμοποιηθούν παρακάτω στην ροή του προγράμματος, για την ολοκλήρωση της περιμένει το σύμβολο ";".
def elsepart()	Διαβάζει το "else" (αν υπάρχει) και καλεί την statements().
def expression()	Υπεύθυνο για την αναγνώριση μίας έκφρασης και την δημιουργία της κατάλληλης quad, με την ολοκλήρωση της, επιστρέφεται η τιμή που αποτίμησε.
def factor()	Επιστρέφει τον παράγοντα που διάβασε, είτε είναι αριθμός, είτε μεταβλητή, είτε έκφραση. Επιστρέφει σφάλμα αν δεν εντοπίσει κάτι από αυτά ή αν δεν έχει κλείσει η παρένθεση της έκφρασης.
def forcaseStat()	Εκτελεί την forcase ελέγχοντας τα condition που βρίσκονται μετά τα case. Μόλις μία από αυτές βρεθεί αληθής, τότε εκτελούνται οι αντίστοιχες statements (που ακολουθούν το condition). Μετά ο έλεγχος μεταβαίνει στην αρχή της forcase. Αν κανένα από τα case δεν ισχύει, τότε ο έλεγχος μεταβαίνει στη default και εκτελούνται οι statements στη default.
def formalparitem()	Υπεύθυνη για την δημιουργία των κατάλληλων entities και arguments για τον πίνακα συμβόλων.
def formalparlist()	Καλεί την formalparitem() όσες φορές είναι απαραίτητο για την ανάγνωση όλων των παραμέτρων της συνάρτησης.
def idtail()	Διαβάζει τις παρενθέσεις και δημιουργεί τον ενδιάμεσο κώδικα που αφορά μία συνάρτηση του προγράμματος που έχουμε δώσει για μετάφραση.
def ifStat()	Εκτελεί την if, φτιάχνοντας τις λίστες conditionTrue και conditionFalse και μία κενή ifList. Διαβάζει το condition και κάνει backpatch για την περίπτωση που είναι αληθές το condition (θα μπει μέσα σε εκείνη την περίπτωση), διαβάζει τα statements και όταν ολοκληρώσει, το ifList θα κοιτά την επόμενη τετράδα. Αν είναι ψευδές το statement, θα συνεχίσει έξω από την if (γι'αυτό και το backpatch

	της false εκεί) όπου θα διαβάσει το else (αν υπάρχει) και θα κάνει και backpatch της ifList με την επόμενη τετράδα.
def incaseStat()	Εκτελεί την incase, ελέγχοντας τα condition που βρίσκονται μετά τα case, εξετάζοντας τα ένα-ένα. Για κάθε μία από αυτές που η αντίστοιχη condition ισχύει, εκτελούνται και τα αντίστοιχα statements (που ακολουθούν το condition). Θα εξεταστούν όλα τα condition και θα εκτελεστούν όλα τα statements των οποίων οι condition ισχύουν. Αφότου εξεταστούν όλα τα case, ο έλεγχος μεταβαίνει έξω από την incase, εάν κανένα από τα statements δεν έχει εκτελεστεί ή μεταβαίνει στην αρχή της incase, έαν έστω και ένα από τα statements έχει εκτελεστεί.
def inputStat()	Διαβάζει το “input” και δημιουργεί τον κατάλληλο ενδιάμεσο κώδικα για την ανάγνωση τιμής από πληκτρολόγιο.
def muloperator()	Διαβάζει τα σύμβολα “*” και “/“.
def optionalSign()	Υπεύθυνη για την αναγνώριση προαιρετικού προσήμου.
def printStat()	Διαβάζει το “print” και δημιουργεί τον κατάλληλο ενδιάμεσο κώδικα για την εμφάνιση τιμής στην οθόνη.
def program()	Διαβάζει την λέξη “program” και το όνομα του προγράμματος μας, δημιουργεί το 1ο scope και καλεί την block(). Περιμένει στο τέλος να βρει “.” και τίποτα παραπάνω.
def reloperator()	Διαβάζει τους χαρακτήρες σύγκρισης και επιστρέφει τον χαρακτήρα που διάβασε, αν δεν βρει χαρακτήρα σύγκρισης, θα πετάξει σφάλμα.
def returnStat()	Διαβάζει το “return” και καλεί την genQuad(“retv”,…) με το expression που διάβασε παραπάνω.
def statement()	Καλεί την κατάλληλη μέθοδο ανάλογα το keyword που αναγνώρισε.
def statements()	Καλεί την statement() και όσες άλλες statement() ακολουθούν στην συνέχεια.
def subprogram()	Υπεύθυνη για την μετάφραση μίας συνάρτησης ή διαδικασίας.
def subprograms()	Καλεί την subprogram() και όσες άλλες subprogram() ακολουθούν στην συνέχεια.
def switchcaseStat()	Εκτελεί την switchcase, ελέγχοντας τα condition που βρίσκονται μετά τα case. Μόλις μία από αυτές βρεθεί αληθής, τότε εκτελούνται τα αντίστοιχα statements (που ακολουθούν το condition) και ο έλεγχος μεταβαίνει έξω από την switchcase. Αν, κατά το πέρασμα, κανένα από τα case δεν ισχύσει, τότε ο έλεγχος μεταβαίνει στην default και εκτελούνται τα default statements.
def term()	Υπεύθυνη για την ανάγνωση των όρων μίας έκφρασης και δημιουργία του αντίστοιχου ενδιάμεσου κώδικα.
def varlist()	Κρατά μία λίστα με όλες τις μεταβλητές για το .c αρχείο και βάζει τα αντίστοιχα entities στον πίνακα συμβόλων.
def whileStat()	Εκτελεί την while, επαναλαμβάνοντας τα statements, όσο το condition ισχύει. Αν την πρώτη φορά που θα αποτιμηθεί η condition, το αποτέλεσμα της αποτίμησης είναι ψευδές, τότε οι statements δεν εκτελούνται ποτέ.

Ενδιάμεσος Κώδικας

Η μετατροπή του κώδικα από την αρχική γλώσσα στην τελική δεν γίνεται απευθείας. Μεσολαβεί η μετατροπή του σε μία ενδιάμεση γλώσσα, την οποία συνηθίζουμε να λέμε και ενδιάμεση αναπαράσταση, η οποία εξακολουθεί να θεωρείται γλώσσα υψηλού επιπέδου, αλλά οι δομές της δεν είναι τόσο σύνθετες, όσο αυτές της αρχικής γλώσσας, ενώ η συντακτική της ανάλυση είναι τετριμμένη.

def genQuad(operator, operand1, operand2, operand3)	Δημιουργεί την επόμενη τετράδα μαζί με τον αριθμό ετικέτας και την προσθέτει στην quadList.
def nextQuad()	Μας επιστρέφει τον αριθμό της επόμενης ετικέτας.
def newTemp()	Μας επιστρέφει το όνομα της επόμενης προσωρινής μεταβλητής που θα χρησιμοποιήσουμε, την προσθέτει στον πίνακα variables και δημιουργεί και το αντίστοιχο entity.
def emptyList()	Μας δημιουργεί και επιστρέφει μια κενή λίστα.
def makeList(x)	Μας δημιουργεί μία λίστα ετικετών τετράδων που περιέχει μόνο το x.
def mergeList(list1, list2)	Μας δημιουργεί μία λίστα ετικετών τετράδων από τη συνένωση των list1 και list2.
def backpatch(listX, labelY)	Η listX αποτελείται από δείκτες σε τετράδες των οποίων το τελευταίο τελούμενο δεν είναι συμπληρωμένο και η backpatch επισκέπτεται μία μία τις τετράδες και τις συμπληρώνει το labelY.
def intercodeGen(file)	Γράφει σε ένα αρχείο κειμένου τον ενδιάμεσο κώδικα με όνομα το όνομα του προγράμματος .ci και κατάληξη .int.
def cCodeGen(file)	Παράγει το ισοδύναμο σε C με όνομα το όνομα του προγράμματος .ci.

Πίνακας Συμβόλων

Ο πίνακας συμβόλων είναι δυναμική δομή στην οποία αποθηκεύεται πληροφορία σχετιζόμενη με τα συμβολικά ονόματα που χρησιμοποιούνται στο υπό μεταγλώττιση πρόγραμμα. Η δομή αυτή παρακολουθεί τη μεταγλώττιση και μεταβάλλεται δυναμικά, με την προσθήκη ή αφαίρεση πληροφορίας σε και από αυτήν, ώστε σε κάθε σημείο της διαδικασίας της μεταγλώττισης να περιέχει ακριβώς την πληροφορία που εκείνη τη στιγμή πρέπει να έχει.

def addEntity(name, type, quad)	Δημιουργεί ένα νέο αντικείμενο τύπου entity αφού πρώτα τσεκάρει αν υπάρχει ήδη αυτό το entity μέσα στο scope. Το entity μπορεί να είναι μεταβλητή, συνάρτηση, διαδικασία, παράμετρος ή προσωρινή μεταβλητή. Προσθέτει το κάθε entity στο κατάλληλο scope και ενημερώνονται διάφορα πεδία, όπως το offset και ο τύπος της παραμέτρου.
def addScope()	Δημιουργεί ένα αντικείμενο τύπου scope και το προσθέτει στον πίνακα συμβόλων (table).
def deleteScope()	Διαγράφει ένα αντικείμενο scope από τον πίνακα συμβόλων, καθώς έχει γίνει η μετάφρασή του και ενημερώνει το τρέχον βάθος.
def addArgument(arg)	Δημιουργεί ένα αντικείμενο τύπου argument και το προσθέτει με μια λίστα από arguments η οποία είναι πεδίο ενός entity(func/proc).
def searchEntity(name)	Αναζητεί το entity με όνομα “name” εντός του πίνακα συμβόλων. Ξεκινάει την αναζήτηση αντίστροφα (από το τρέχον scope προς τα ανώτερα) και επιστρέφει το πρώτο που θα συναντήσει. Αν δεν το βρεί, θα τυπώσει μήνυμα σφάλματος και θα τερματιστεί η εκτέλεση του μεταφραστή.
def existEntity(entityName, entityType)	Κάθε φορά που καλείται μία συνάρτηση ή διαδικασία στο πρόγραμμα .ci, καλούμε την existEntity ώστε να ελέγχουμε αν υπάρχει στον πίνακα συμβόλων η κληθείσα συνάρτηση ή διαδικασία. Σε περίπτωση που δεν

	βρει κάτι στον πίνακα συμβόλων, θα τυπώσει μήνυμα σφάλματος και θα τερματιστεί η εκτέλεση του μεταφραστή.
def symbolTableGen()	Μόλις τελειώσουμε με ένα block και πριν διαγράψουμε κάποιο scope, καλείται η symbolTableGen ώστε να αποθηκεύσουμε σε μια λίστα (symTable) σε μορφή string, την παρούσα κατάσταση του πίνακα συμβόλων.
def writeSymTable(Filename)	Διατρέχοντας τη λίστα που έχει strings με πληροφορίες από τον πίνακα συμβόλων, γράφει σε ένα αρχείο κειμένου τον πίνακα συμβόλων με όνομα το όνομα του προγράμματος .ci και κατάληξη .symb.

Τελικός Κώδικας

Η τελευταία φάση της παραγωγής κώδικα είναι η παραγωγή του τελικού κώδικα. Ο τελικός κώδικας προκύπτει από τον ενδιάμεσο κώδικα με τη βοήθεια του πίνακα συμβόλων. Συγκεκριμένα, από κάθε εντολή ενδιάμεσου κώδικα προκύπτει μία σειρά εντολών τελικού κώδικα, η οποία για να παραχθεί ανακτά πληροφορίες από τον πίνακα συμβόλων.

def gnlvcode(variable)	Δημιουργεί τελικό κώδικα για την προσπέλαση πληροφορίας που βρίσκεται αποθηκευμένη στο εγγράφημα δραστηριοποίησης κάποιου προγόνου της συνάρτησης ή της διαδικασίας που αυτή τη στιγμή μεταφράζεται. Δηλαδή μεταφέρει στον καταχωρητή t0 τη διεύθυνση μιας μη τοπικής μεταβλητής. Από τον πίνακα συμβόλων βρίσκει πόσα επίπεδα επάνω βρίσκεται η μη τοπική μεταβλητή.
def loadvr(v, reg)	Παράγει τελικό κώδικα ο οποίος διαβάζει μία μεταβλητή που είναι αποθηκευμένη στη μνήμη και την μεταφέρει σε έναν καταχωρητή. Δηλαδή μεταφέρει δεδομένα από τη μεταβλητή v στον καταχωρητή reg, ενώ για να δώσει τις σωστές εντολές assembly, λαμβάνει υπόψη της το αν η μεταβλητή v είναι καθολική, τοπική, τυπική παράμετρος, καθώς και τον τρόπο περάσματος (με τιμή ή με αναφορά).
def storerv(reg, v)	Κάνει την αντίστροφη διαδικασία από το loadvr, παράγει τελικό κώδικα ο οποίος αποθηκεύει στη μνήμη την τιμή μιας μεταβλητής η οποία βρίσκεται σε έναν καταχωρητή. Φροντίζει για τη σωστή αποθήκευση του καταχωρητή reg στην μεταβλητή v. Έχει διαφορετική συμπεριφορά ανάλογα με τον αν έχουμε καθολική μεταβλητή, τοπική μεταβλητή ή τυπική παράμετρο, καθώς και το βάθος φωλιάσματος.
def riscVBlock(blockName, sQuad)	Ως παραμέτρους δέχεται το sQuad ενός μπλοκ (δηλαδή την $1^{η}$ τετράδα του μπλοκ) και το όνομα του (blockName). Για κάθε έτοιμο (στον ενδιάμεσο κώδικα) μπλοκ, η συνάρτηση riscVBlock, διατρέχει την λίστα στην οποία έχουμε αποθηκεύσει τα quads και για κάθε quad καλεί την riscVAssemblyGen().
def riscVAssemblyGen(quad, blockName)	Δέχεται ως παραμέτρους το όνομα ενός μπλοκ και μία τετράδα ενδιάμεσου κώδικα. Σε κάθε περίπτωση, ανάλογα με τον τελεστή (operator=quad[1]) που διαβάζει, προσθέτει κατάλληλες εντολές assembly για riscV σε μια λίστα τελικού κώδικα (final). • Αν εντοπίσει “jump”, θα φροντίσει για την δημιουργία του κατάλληλου branch στον τελικό κώδικα.

	• Αν εντοπίσει “+” ή “-” ή “*” ή “/”, θα φορτώσει τις μεταβλητές σε 2 προσωρινούς καταχωρητές, θα εκτυπώσει την κατάλληλη εντολή τελικού κώδικα και θα αποθηκεύσει το αποτέλεσμα σε ένα καταχωρητή.
	• Αν εντοπίσει “=” ή “<>” ή “>” ή “<” ή “>=” ή “<=”, θα φορτώσει τις μεταβλητές σε 2 προσωρινούς καταχωρητές και θα εκτυπώσει την κατάλληλη εντολή τελικού κώδικα για το αντίστοιχο είδος σύγκρισης.
	• Αν εντοπίσει “:=”, θα φορτώσει και θα αποθηκεύσει σε μία προσωρινή μεταβλητή την ανάθεση που γίνεται.
	• Αν εντοπίσει “out”, θα τυπώσει την τιμή που έχει ο καταχωρητής.
	• Αν εντοπίσει “inp”, θα ζητήσει τιμή από τον χρήστη και την βάζει στον καταχωρητή.
	• Αν εντοπίσει “retv”, θα προετοιμάσει μία μεταβλητή για την επιστροφή και θα εκτυπώσει τις κατάλληλες εντολές τελικού κώδικα.
	• Αν εντοπίσει “halt”, θα εκτυπώσει τις κατάλληλες εντολές τελικού κώδικα για τερματισμό του προγράμματος.
	• Αν εντοπίσει “begin_block”, αναλόγως αν το μπλοκ είναι κυρίου προγράμματος ή υποπρογράμματος, θα εκτυπώσει τις κατάλληλες εντολές τελικού κώδικα.
	• Αν εντοπίσει “end_block”, αν αφορά υποπρόγραμμα, θα εκτυπώσει τις κατάλληλες εντολές τελικού κώδικα.
	• Αν εντοπίσει “par” στον ενδιάμεσο κώδικα, διαβάζουμε τα επόμενα quads μέχρι να βρούμε “call”. Σε μια λίστα κρατάμε αρίθμηση παραμέτρων (paramList). Η παραπάνω ενέργεια γίνεται επειδή χρειαζόμαστε το framelength του υποπρογράμματος (scope) που πρόκειται να καλεστεί (calleeEntity.offset). Χρησιμοποιούμε την paramFlag ώστε να γνωρίζουμε αν είμαστε εντός της παραπάνω κατάστασης ή όχι. Για κάθε τύπο παραμέτρου που διαβάζουμε (cv, ref, ret), διακρίνουμε περιπτώσεις ώστε πάντα να βρίσκουμε τον σωστό caller και callee και εκτυπώνουμε τον κατάλληλο τελικό κώδικα.
	  ○ Αν έχουμε “cv”, απλά θα αντιγράψουμε την τιμή.
	  ○ Αν έχουμε “ref”, ανάλογα με το scope που βρίσκουμε το entity και το είδος του, πράττουμε ανάλογα για να πάρουμε την τιμή με αναφορά από την παράμετρο/μεταβλητή.
	  ○ Αν έχουμε “ret”, αφορά τιμή που θα επιστρέψει μία συνάρτηση.
	• Αν εντοπίσει “call” στον ενδιάμεσο κώδικα, Από τον πίνακα συμβόλων διαβάζουμε το scope των caller και callee, και για κάθε περίπτωση, εκτυπώνουμε και τον κατάλληλο τελικό κώδικα.

def writeRiscVAsm(file)	Διατρέχοντας τη λίστα final που έχει string τελικού κώδικα, γράφει σε ένα αρχείο κειμένου τον τελικό κώδικα με όνομα το όνομα του προγράμματος .ci και κατάληξη .asm.