mlc++
ένας απλός μεταφραστής για MIPS

Μάιος 2020

Περιεχόμενα

1 Η γλώσσα miminal++ **3**
- 1.1 Εισαγωγή . . . 3
- 1.2 Λεκτικές Μονάδες . . . 3
- 1.3 Μορφή προγράμματος . . . 4
- 1.4 Τύποι και δηλώσεις μεταβλητών . . . 5
- 1.5 Τελεστές και εκφράσεις . . . 5
- 1.6 Δομές τις γλώσσας . . . 5
  - 1.6.1 Εκχώρηση . . . 5
  - 1.6.2 Απόφαση if . . . 6
  - 1.6.3 Επανάληψη while . . . 6
  - 1.6.4 Επανάληψη loop . . . 6
  - 1.6.5 Επανάληψη forcase . . . 6
  - 1.6.6 Επανάληψη incase . . . 7
  - 1.6.7 Επανάληψη doublewhile . . . 7
  - 1.6.8 Επιστροφή τιμής συνάρτησης . . . 8
  - 1.6.9 Έξοδος δεδομένων . . . 8
  - 1.6.10 Είσοδος δεδομένων . . . 8
  - 1.6.11 Κλήση διαδικασίας . . . 8
  - 1.6.12 Έξοδος από βρόχο loop . . . 8
- 1.7 Υποπρογράμματα . . . 9
- 1.8 Μετάδοση παραμέτρων . . . 10

2 Χρήση του μεταφραστή **11**

3 Λεκτική Ανάλυση **12**
- 3.1 Πεπερασμένο Αυτόματο . . . 12
- 3.2 Λεκτικός Αναλυτής . . . 13

4 Συντακτική Ανάλυση **14**

5 Ενδιάμεση γλώσσα **16**
5.1 Εισαγωγή 16
5.2 Εκφράσεις 17
5.3 Συνθήκες 18
5.4 if - else 19
5.5 while 19
5.6 loop 19
5.7 forcase 20
5.8 incase 20
5.9 doublewhile 20

6 Πίνακας Συμβόλων **22**

7 Errors **25**

8 Παραγωγή κώδικα assembly MIPS **26**

9 Δημιουργία κώδικα προσομοίωσης C **28**

Κεφάλαιο 1

Η γλώσσα miminal++

1.1 Εισαγωγή

Η minimal++ είναι μια απλή και μικρή γλώσσα προγραμματισμού η οποία παράγει κώδικα assembly για επεξεργαστές βασισμένους στην αρχιτεκτονική MIPS. Παρόλο που οι προγραμματιστικές της ικανότητες είναι μικρές, η εκπαιδευτική αυτή γλώσσα περιέχει πλούσια στοιχεία και η κατασκευή του μεταγλωττιστή της έχει να παρουσιάσει αρκετό ενδιαφέρον, αφού περιέχονται σε αυτήν πολλές εντολές που χρησιμοποιούνται από άλλες γλώσσες, καθώς και κάποιες πρωτότυπες. Η minimal++ υποστηρίζει συναρτήσεις και διαδικασίες, μετάδοση παραμέτρων με αναφορά και τιμή, αναδρομικές κλήσεις και άλλες ενδιαφέρουσες δομές. Επίσης, επιτρέπει φώλιασμα στη δήλωση συναρτήσεων κάτι που λίγες γλώσσες υποστηρίζουν (το υποστηρίζει η Pascal, δεν το υποστηρίζει η C).Από την άλλη όμως πλευρά, η minimal++ δεν υποστηρίζει βασικά προγραμματιστικά εργαλεία όπως η δομή for, ή τύπους δεδομένων όπως οι πραγματικοί αριθμοί και οι συμβολοσειρές.

1.2 Λεκτικές Μονάδες

Το αλφάβητο της minimal++ αποτελείται από:
• τα μικρά και κεφαλαία γράμματα της λατινικής αλφαβήτου («A»,...,«Z» και «a»,...,«z»),
• τα αριθμητικά ψηφία («0»,...,«9»),
• τα σύμβολα των αριθμητικών πράξεων («+», «-», «*», «/»),
• τους τελεστές συσχέτισης «<», «>», «=», «<=», «>=», «<>»,
• το σύμβολο ανάθεσης «:=»,
• τους διαχωριστές («;», «,», «:»)

• καθώς και τα σύμβολα ομαδοποίησης («(»,«)»,«[» ,«]»,««»,«»»)

Τα σύμβολα ”[” και ”]” χρησιμοποιούνται στις λογικές παραστάσεις όπως τα σύμβολα ”(” και ”)” στις αριθμητικές παραστάσεις.

Οι δεσμευμένες λέξεις είναι:

program , **declare** , **if** , **else** , **while** , **doublewhile** , **loop** , **exit** , **forcase** , **incase** , **when** , **default** , **not** , **and** , **or** , **function** , **procedure** , **call** , **return** , **in** , **inout** , **input** , **print**

Οι λέξεις αυτές δεν μπορούν να χρησιμοποιηθούν ως μεταβλητές. Οι σταθερές της γλώσσας είναι ακέραιες σταθερές που αποτελούνται από προαιρετικό πρόσημο και από μία ακολουθία αριθμητικών ψηφίων.Τα αναγνωριστικά της γλώσσας είναι συμβολοσειρές που αποτελούνται από γράμματα και ψηφία,αρχίζοντας όμως από γράμμα. Ο μεταγλωττιστής λαμβάνει υπόψη του μόνο τα τριάντα πρώταγράμματα. Οι λευκοί χαρακτήρες (tab, space, return) αγνοούνται και μπορούν να χρησιμοποιηθούν με οποιονδήποτε τρόπο χωρίς να επηρεάζεται η λειτουργία του μεταγλωττιστή, αρκεί βέβαια να μην βρίσκονται μέσα σε δεσμευμένες λέξεις, αναγνωριστικά, σταθερές. Το ίδιο ισχύει και για τα σχόλια,τα οποία πρέπει να βρίσκονται μέσα στα σύμβολα /* και */ ή να βρίσκονται μετά το σύμβολο // και ως το τέλος της γραμμής. Απαγορεύεται να ανοίξουν δύο φορές σχόλια, πριν τα πρώτα κλείσουν. Δεν υποστηρίζονται εμφωλευμένα σχόλια.

1.3 Μορφή προγράμματος

```
program id
{
      declarations
      subprograms
      sequence of statements
}
```

1.4 Τύποι και δηλώσεις μεταβλητών

Ο μοναδικός τύπος δεδομένων που υποστηρίζει η minimal++ είναι οι ακέραιοι αριθμοί. Οι ακέραιοι αριθμοί πρέπει να έχουν τιμές από –32767 έως 32767. Η δήλωση γίνεται με την εντολή declare.Ακολουθούν τα ονόματα των αναγνωριστικών χωρίς καμία άλλη δήλωση, αφού γνωρίζουμε ότι πρόκειται για ακέραιες μεταβλητές και χωρίς να είναι αναγκαίο να βρίσκονται στην ίδια γραμμή. Οι μεταβλητές χωρίζονται μεταξύ τους με κόμματα. Το τέλος της δήλωσης αναγνωρίζεται με το ελληνικό ερωτηματικό. Επιτρέπεται να έχουμε περισσότερες των μία συνεχόμενες χρήσεις της declare.

1.5 Τελεστές και εκφράσεις

Η προτεραιότητα των τελεστών από τη μεγαλύτερη στη μικρότερη είναι:
(1) Μοναδιαίοι λογικοί: «not»
(2) Πολλαπλασιαστικοί: «*», «/»
(3) Μοναδιαίοι προσθετικοί: «+», «-»
(4) Δυαδικοί προσθετικοί: «+», «-»
(5) Σχεσιακοί «=», «<», «>», «<>», «<=», «>=»
(6) Λογικό «and»
(7) Λογικό «or»

1.6 Δομές τις γλώσσας

1.6.1 Εκχώρηση

Id := expression
Χρησιμοποιείται για την ανάθεση της τιμής μιας μεταβλητής ή μιας σταθεράς, ή μιας έκφρασης σε μία μεταβλητή.

1.6.2 Απόφαση if

if(condition)
statements1
[else
statements2]
Η εντολή απόφασης if εκτιμάει εάν ισχύει η συνθήκη condition και εάν πράγματι ισχύει,τότε εκτελούνται οι εντολές statements1 που το ακολουθούν. Το else δεν αποτελεί υποχρεωτικό τμήμα της εντολής και γι' αυτό βρίσκεται σε αγκύλη. Οι εντολές statements2 που ακολουθούν το else εκτελούνται εάν η συνθήκη condition δεν ισχύει

1.6.3 Επανάληψη while

while(condition)
statements
Η εντολή επανάληψης while επαναλαμβάνει συνεχώς τις εντολές statements, όσο η συνθήκη condition ισχύει. Αν την πρώτη φορά που θα αποτιμηθεί η condition, το αποτέλεσμα της αποτίμησης είναι ψευδές, τότε οι statements δεν εκτελούνται ποτέ.

1.6.4 Επανάληψη loop

loop
statements
Η εντολή επανάληψης loop επαναλαμβάνει για πάντα τις εντολές statements. Έξοδος από το βρόχο γίνεται μόνο όταν κληθεί η εντολή **exit**.

1.6.5 Επανάληψη forcase

forcase
(when:(condition): statements1)*
default: statements2
Η δομή επανάληψης forcase ελέγχει τις condition που βρίσκονται μετά

τα when. Μόλις μία από αυτές βρεθεί αληθής, τότε εκτελούνται οι statements1 που ακολουθούν. Μετά ο έλεγχος μεταβαίνει στην αρχή της forcase. Αν καμία από τις when δεν ισχύει, τότε ο έλεγχος μεταβαίνει στη default και εκτελούνται οι statements2. Στη συνέχεια ο έλεγχος μεταβαίνει έξω από την forcase.

1.6.6 Επανάληψη incase

incase
(when:(condition): statements)*
Η δομή επανάληψης incase ελέγχει τις condition που βρίσκονται μετά τα when, εξετάζοντας τες κατά σειρά. Για κάθε μία από αυτές που η αντίστοιχη condition ισχύει, εκτελούνται οι statements που ακολουθούν το σύμβολο ":". Θα εξεταστούν όλες οι condition και θα εκτελεστούν όλες οι statements των οποίων οι condition ισχύουν. Αφότου εξεταστούν όλες οι when, ο έλεγχος μεταβαίνει έξω από τη δομή incase εάν καμία από τις statements δεν έχει εκτελεστεί ή μεταβαίνει στην αρχή της incase, έαν έστω και μία από τις statements έχει εκτελεστεί.

1.6.7 Επανάληψη doublewhile

doublewhile(condition)
statements1
else
statements2
Την πρώτη φορά που ο έλεγχος εισέρχεται στον βρόχο doublewhile, αποφασίζεται μέσα από την condition αν η εκτέλεση θα μεταβεί στο statements1 (true) ή αν θα μεταβεί στο statements2 (false). Από το statements1 φεύγει, όταν η συνθήκη σταματήσει να είναι true.Από το statements2 φεύγει όταν η συνθήκη σταματήσει να είναι false. Και στις δύο αυτές περιπτώσεις ο έλεγχος μεταφέρεται έξω από την δομή. Δηλαδή, δεν είναι ποτέ δυνατόν σε μία εκτέλεση της doublewhile ο έλεγχος να περάσει και από την statements1 και από την statements2.
Χρησιμοποιείται μέσα σε συναρτήσεις για να επιστρέφει το αποτέλεσμα της συνάρτησης.

1.6.8 Επιστροφή τιμής συνάρτησης

return (expression)
Χρησιμοποιείται μέσα σε συναρτήσεις για να επιστρέφει το αποτέλεσμα της συνάρτησης.

1.6.9 Έξοδος δεδομένων

print (expression)
Εμφανίζει στην οθόνη το αποτέλεσμα της αποτίμησης του expression.

1.6.10 Είσοδος δεδομένων

input (id)
Ζητάει από τον χρήστη να δώσει μια τιμή μέσα από το πληκτρολόγιο.

1.6.11 Κλήση διαδικασίας

call function_name(actual_parameters)
Καλεί μια διαδικασία.

1.6.12 Έξοδος από βρόχο loop

exit
Εκτελεί έξοδο από βρόχο loop.

1.7 Υποπρογράμματα

Η minimal++ υποστιρίζει συναρτήσεις.
function id(formal_pars)
{
    declarations
    subprograms
    statements
}
Η formal_pars είναι η λίστα των τυπικών παραμέτρων. Οι συναρτήσεις μπορούνν να φωλιάσουν η μία μέσα στην άλλη και οι κανόνες εμβέλειας είναι όπως της PASCAL. Η επιστροφή της τιμής μιας συνάρτησης γίνεται με την return.Η κλήση μιας συνάρτησης, γίνεται από τις αριθμητικές παραστάσεις σαν τελούμενο. π.χ.
D = a + f(**in** x)
όπου f η συνάρτηση και x παράμετρος που περνάει με τιμή. Οι διαδικασίες συντάσσονται ως εξής:
procedure id(formal_pars)
{
    declarations
    subprograms
    statements
}
Η κλήση μιας διαδικασίας, γίνεται με την call.π.χ.
call f(**inout** x)
όπου f η διαδικασία και x η παράμετρος που περνάει με αναφορά.

Μια συνάρτηση μπορεί να έχει απευθείας πρόσβαση εκτός από τις μεταβλητές της και στις μεταβλητές των προγόνων της, σε περίπτωση ίδιων αναγνωριστικών, προτεραιότητα έχουν οι μεταβλητές της συνάρτησης και μετά του κοντινότερου προγόνου της. Όλες οι συναρτήσεις έχουν απευθείας πρόσβαση στις μεταβλητές του κύριου προγράμματος.
Μια συνάρτηση μπορεί να καλέσει τον εαυτό της τα εμφολευμένα παιδία συναρτήσεις της και οποιοδήποτε από τους προγόνους της έως το 1ο βάθος φωλιάσματος. Μετά μπορεί να καλέσει όσες συναρτήσεις έχουν δηλωθεί πιο πριν στο 1ο βάθος φωλιάσματος.
Για να δημιουργήσουμε μια συνάρτηση, δεν πρέπει να υπάρχει άλλη συνάρτηση στο ίδιο βάθος φωλιάσματος με το ίδιο όνομα, τον ίδιο τύπο και τα ίδια ορίσματα, αν έστω ένα από τα παραπάνω δεν ισχύει τότε η συνάρτηση μπορεί να δημιουργηθεί.
Εάν μια συνάρτηση(καλούσα), η πρόγονος συνάρτηση και το παιδί συ-

νάρτηση της έχουν το ίδιο όνομα, ίδιο τύπο και τα ίδια ορίσματα τότε η προτεραιότητα όταν γίνεται μια κλήση συνάρτησης είναι:
1) παιδί συνάρτηση
2) καλούσα συνάρτηση
3) πρόγονος συνάρτηση

1.8 Μετάδοση παραμέτρων

Η minimal++ υποστηρίζει δύο τρόπους μετάδοσης παραμέτρων:
• με σταθερή τιμή. Δηλώνεται με την λεκτική μονάδα in. Αλλαγές στην τιμή της δεν επιστρέφονται στο πρόγραμμα που κάλεσε τη συνάρτηση.
• με αναφορά. Δηλώνεται με τη λεκτική μονάδα inout. Κάθε αλλαγή στη τιμή της μεταφέρεται αμέσως στο πρόγραμμα που κάλεσε τη συνάρτηση.Στην κλήση μίας συνάρτησης οι πραγματικοί παράμετροι συντάσσονται μετά από τις λέξεις κλειδιά in και inout, ανάλογα με το αν περνάνε με τιμή ή αναφορά.

Κεφάλαιο 2

Χρήση του μεταφραστή

Ο μεταφραστής ονομάζεται mlc και ο κώδικας του βρίσκεται στο ομώνυμο αρχείο με κατάληξη .py. Για να τρέξει ο μεταφραστής η γλώσσα python 3 χρειάζεται να είναι εγκατεστημένη στο σύστημα. Η χρήση του μεταφραστή είναι απλή :
`python3 mlc.py option file.min`
όπου file.min το αρχείο που περιέχει τον κώδικα του προγράμματος σε γλώσσα minimal++ και option μια παράμετρος.
`option:`
`--help` : εμφανίζει ένα κείμενο βοήθειας για τον μεταφραστή.
`-save-temps` : αποθηκεύει τα προσωρινά αρχεία που χρησιμοποιεί ο μεταφραστής.Συγκεκριμένα το αρχείο με την ενδίαμεση γλώσσα,το αρχείο με πληροφορίες για τον πίνακα συμβόλων και το αρχείο για την προσομοίωση του κώδικα σε C.
επίσης μπορεί να μην μπει καμία παράμετρος και έτσι ο μεταφραστής θα παράγει απευθείας το αρχείο της assembly MIPS με κατάληξη .asm. Όλα τα προγράμματα της minimal++ πρέπει να είναι σε αρχεία με κατάληξη .min .

Κεφάλαιο 3

Λεκτική Ανάλυση

3.1 Πεπερασμένο Αυτόματο

Το πεπερασμένο αυτόματο(ΠΑ) απότελειται από τις κλάσεις **State**, **Symbols**, **Id** καθώς και από τον πίνακα κατακερματισμού **automata_states**. Οι κλάση State δεν είναι μια κλάση με την έννοια του αντικειμενοστραφή προγραμματισμού αλλά χρησιμοποιείται κυρίως για απαρίθμηση των καταστάσεων του ΠΑ. Το ίδιο ισχύει και για την κλάση Id η οποία χρησιμοποιείται για την απαρίθμηση των συμβόλων της γλώσσας.Τέλος η Symbols περιέχει όλες τις λεκτικές μονάδες της γλώσσας, και στις 3 αυτές περιπτώσεις χρησιμοποιήθηκαν κλάσεις μιας και η python μας δίνει αυτή την δυνατότητα απαρίθμηση και αποθήκευση πεδίων.
Το **automata_states** αποτελεί το σημαντικότερο κομμάτι του ΠΑ μιας και είναι ένας πίνακας κατακερματισμού που περιέχει όλες τις καταστάσεις και τις καταστάσεις στις οποίες μια κατάσταση μπορεί να πάει. Κάθε στοιχείο του πίνακα είναι μια κατάσταση η οποία είναι το κλειδί και συνδέεται με μια λίστα η οποία έχει το ρόλο της τιμής και περιέχει πληροφορίες για τις καταστάσεις που μπορεί να πάει η κατάσταση. Συγκεκριμένα το πεδίο **next_state** περιέχει της επόμενη κατάσταση, το πεδίο **condition** περιέχει την συνθήκη για την εναλλαγή των καταστάσεων, το πεδίο **go_back** περιέχει μια boolean τιμή η οποία διασφαλίζει αν οι δυο καταστάσεις είναι αμφίδρομες(True) ή αν μόνο από την μια κατάσταση μπορούμε να πάμε στην άλλη(False). Τέλος το πεδίο **id** το έχουν ορισμένες καταστάσεις είναι το αναγνωριστικό του τρέχοντος συμβόλου, το αναγνωριστικό αυτό είναι μέρος της κλάσης Id.

3.2 Λεκτικός Αναλυτής

Ο λεκτικός αναλυτής βρίσκεται στην κλάση **lex**.Στον constructor του παίρνει σαν όρισμα το αρχείο με το πρόγραμμα σε minimal++. Η συνάρτηση **next_char** διαβάζει τον επόμενο χαρακτήρα από το αρχείο, αποθηκεύει την θέση του(file_index) και τον επιστρέφει, αν ο χαρακτήρας είναι ο χαρακτήρας νέας γραμμής τότε αυξάνει τον μετρητη γραμμών(file_line) κατά ένα. Η συνάρτηση **undo_read** επιστρέφει στην αρχή της λέξης που διάβαστηκε και αφαιρεί από τον μετρητή γραμμών τον κατάλληλο αριθμό γραμμών, χρησιμοποιείται στις περιπτώσεις που δεν ξέρουμε τις ακριβώς να αναμένουμε σαν επόμενη λέξη. Η συνάρτηση **start_read** ξεκινάει την διαδικασία εύρεσης της επόμενης λέξης χρησιμοποιώντας το πεπερασμένο αυτόματο και μέχρι να φτάσει στην τελική κατάσταση. Στην περίπτωση των σχολίων μέσα στο πρόγραμμα η start_read τα αγνοεί και συνεχίζει στην επόμενη λέξη.

Κεφάλαιο 4

Συντακτική Ανάλυση

Ο συντακτικός αναλυτής είναι ίσως το πιο σημαντικό κομμάτι του μεταφραστή mlc. Ο συντακτικός αναλυτής αρχικά επιτελεί την κύρια δουλειά του η οποία είναι να εξομοιώσει την γραμματικη της γλώσσας, δηλαδή ξεκινώντας από την συνάρτηση **program** καλούνται οι υπόλοιπες συναρτήσεις όπως ακριβώς γίνεται η μετάβαση στην γραμματική της γλώσσας. Σε κάθε σηνάρτηση του συντακτικού αναλυτή γίνεται έλεγχος αν η λέξη που δόθηκε από τον λεκτικό αναλυτή είναι η αναμενώμενη, αυτός ο έλεγχος γίνεται μέσω του (error_handler)(Errors) και εμφανίζεται το ανάλογο σφάλμα. Π.χ:
Στις πρώτες γραμμές της συνάρτησης **program** βλέπουμε: *word, ID = self.lex.start_read()*
self.error_handler.error_handle(error_types.SyntaxCheckWordIdFatal, "program", Id.IDENTIFIER,word, ID)
Εδώ τα word, ID παίρνουν την λέξη και το είδος της αντίστοιχα από τον λεκτικό αναλυτή έπειτα η πρώτη λέξη που περιμένουμε σε κάθε πρόγραμμα (εκτός των σχολίων που αγνοεί ο λεκτικός) είναι η λέξη program σύμφωνα με την γραμματική της γλώσσας. Έτσι ο error_handler παίρνει σαν πρώτο όρισμα τον τύπο του σφάλματος που είναι SyntaxCheckWordIdFatal το οποίο εμφανίζει σφάλμα αν η λέξη word δεν είναι "program" ή αν το ID δεν είναι τύπου IDENTIFIER.
Στον constructor του συντακτικού αναλύτή αρχικά δημιουργούνται τα αντικείμενα για τα λάθη(error_handler), τον λεκτικό αναλυτή(lex), την ενδιάμεση γλώσσα(inLan) και του πίνακα συμβόλων. Έπειτα καλείται η program και σειρά παίρνει ο έλεγχος της γραμματικής μαζί με την ταυτόχρονη δημιουργία της ενδιάμεσης γλώσσας και του πίνακα συμβόλων. Μόλις τελειώσει ο έλεγχος έχουν σχηματιστεί τα αρχεία της ενδιάμεσης γλώσσας και του πίνακα συμβόλων(δεν χρειάζεται κάπου) και δημιουργείται το αντικείμενο mip_ass το οποίο δημιουργεί το αρχείο με

τον κώδικα assembly MIPS από το αρχείο της ενδιάμεσης γλώσσας και τον πίνακα συμβόλων. Τέλος εάν η παράμετρος -save-temps έχει περαστεί δημιουργείται και το αρχείο με τον κώδικα σε C για testing, εάν δεν περαστεί η παράμετρος αυτή δεν δημιουργείται το αρχείο αυτό και τα αρχεία της ενδιάμεσης γλώσσας και του πίνακα συμβόλων διαγράφονται.

Κεφάλαιο 5

Ενδιάμεση γλώσσα

5.1 Εισαγωγή

Το αντικείμενο **inLan** που δημιουργείται στον constructor του συντακτικού αναλυτή είναι αυτό που θα δημιουργήσει τις τετράδες της ενδιάμεσης γλώσσας για το πρόγραμμα. Αρχικά η **int_lang** είναι η κλάση του αντικειμένου inLan, στον constructor της δημιουργεί το αρχείο που θα γραφτούν οι τετράδες και θέτει κάποιες μεταβλήτες οι οποίες χρησιμοποιούνται για ελέγχους, κυρίως όμως αρχικοποιεί την **function_list** η οποία είναι μια λίστα με όλες τις συναρτήσεις που έχουν διαβαστεί ως εκείνη την στιγμή. Η **function_list** αποθηκεύει της συναρτήσεις με την σειρά που διαβάζονται, έτσι πρώτο πάντα θα είναι το κύριο πρόγραμμα και θα ακουλουθεί η συνάρτηση του 1ο βάθους φωλιάσματος (αν υπάρχει) και όλα τα παίδια και τα παιδία συναρτήσεις τους(οι εμφολευμένες) με την τελευταία εμφολευμένη συνάρτηση να είναι στην ουσία η πρώτη για της οποία θα δημιουργηθούν οι τετράδες της ενδιάμεσης γλώσσας μιας και είναι η πρώτη που θα διαβαστούν τα statements της. Έπειτα γυρνώντας προς τα πίσω διαβάζονται και τα statements των προηγούμενων συναρτήσεων και δημιουργούνται οι αντίστιχες τετράδες μέχρι να επιστρέψουμε στο 1ο βάθος φωλιάσματος όπου και θα διαβάσουμε την επόμενη συνάρτηση(αν υπάρχει) στο 1ο βάθος φωλίασματος και θα συνεχίσουμε ανάλογα. Κάθε φορά που μια συνάρτηση φτάνει στο τέλος της, γράφεται στο αρχείο της ενδιάμεσης γλώσσας και διαγράφεται από την **function_list** μιας και δεν χρειάζεται πλέον.
Η συνάρτηση **relative_function_pos** επιστρέφει την θέση της τρέχουσας συνάρτησης, ενώ η συνάρτηση **nextquad** επιστρέφει τον αριθμό της επόμενης τετράδας. Η συνάρτηση **make_list** σημιουργεί τις πρώτες τετράδες μιας νέας συνάρτησης και την αποθηκεύει στην function_list.

Η συνάρτηση **write_list** γράφει όλες τις τετράδες της τρέχουσας συνάρτησης στο αρχείο της ενδιάμεσης γλώσσας, αν η συνάρτηση είναι το κύριο πρόγραμμα τότε καλεί την συνάρτηση **write_first_line** η οποία βάζει στην πρώτη γραμμή του αρχείου της ενδιάμεσης γλώσσας ένα jump στον αριθμό της τετράδας του κύριου προγράμματος.
Η συνάρτηση **genquad** δημιουργεί της επόμενη τετράδα για την τρέχουσα συνάρτηση. Η συνάρτηση **get_condition** επιστρέφει όλο τον κώδικα για ένα κομμάτι μιας συνθήκης, όταν παρθούν όλα τα κομμάτια του κώδικα τότε καλείται η **backpatch** η οποία συμπληρώνει τις μη συμπληρωμένες τετράδες της συνθήκης(τα jump και relational operators]) έπειτα καλείται η συνάρτηση **add_condition** η οποία ξανά εισάγει τον κώδικα των τετράδων της συνθήκης στο αρχείο.
Η συνάρτηση **backpatch** συμπληρώνει τις τετράδες μιας συνθήκης. Συγκεκριμένα βρίσκει τα άλματα ή τους σχεσιακούς τελεστές(οι και τα δύο ανάλογα το mode) και θέτει σε ποια τετράδα θα πρέπει να γίνει το άλμα τους σε περιπτώση που η συνθήκη είναι αληθής αλλά και στην περίπτωση που είναι ψευδής.
Η συνάρτηση **newtemp** δημιουργεί μια νέα προσωρινή μεταβλητή, ενώ η συνάρτηση **reset_newtemp** ξεκινάει την αρίθμηση των προσωρινών μεταβλητών από την αρχή. Η συνάρτηση **delete** διαγράφει το αρχείο της ενδιάμεσης γλώσσας ενώ η συνάρτηση **close** το αποθηκεύει. Η συνάρτηση **reverse_relop** χρησιμοποιήται στην περίπτωση του not στον κώδικα και αντιστρέφει τον σχεσιακό τελεστή ώστε να γίνει άλμα στην περίπτωση που ισχύει η το αντίστροφο(not). Η συνάρτηση **isInt** χρησιμοποιείται για να διαπιστωθεί εάν ένα όρισμα στην κλήση μιας συνάρτησης είναι ακέραιος.
Η συνάρτηση **special_loop** χρησιμοποιείται στον κώδικα για την δομή loop, συγκεκριμένα βρίσκει την τετράδα exit στο κώδικα ενδιάμεσης γλώσσας που παράχθηκε για το loop και την μετασχηματίζει σε jump στην επόμενη τετράδα μετά τον κώδικα του loop. Παρομοίως η συνάρτηση **special_doublewhile** βρίσκει την τετράδα της συνθήκης του doublewhile που κάνει jump στο κομμάτι true(και μετά false) και θέτει την διεύθυνση της τετράδας που θα κάνει άλμα.
Ακουλουθούν οι δομές και η υλοποίηση τους σε ενδιάμεση γλώσσα.

5.2 Εκφράσεις

Η δημιουργία εκφράσεων σε ενδιάμεση γλώσσα πραγματοποιείται από ένα πλήθος συναρτήσεων που ανήκουν στην γραμματική της γλώσ-

σας. Αρχικά η συνάρτηση **factor** βρίκει εάν ένα μέρος της έκφρασης είναι σταθερά ή μεταβλητή ή προσωρινή μεταβλητή ή κλήση συνάρτησης και επιστρέφει ότι βρει στην συνάρτηση **term**. Η συνάρτηση term με την σειρά της εκχωρεί αυτό που της γύρισε η factor σε μια προσωρινή μεταβλητή εάν η πράξη που ακουλουθεί στην έκφραση είναι πολλαπλασιασμός(*) ή διαίρεση(/), σε αντίθετη περίπτωση επιστρέφει αυτό που πείρε από την factor στην **expression**. Η expression με την σειρά της εκχωρεί αυτό που της γύρισε η term σε μια προσωρινή μεταβλητή εάν η πράξη που ακουλουθεί στην έκφραση είναι πρόσθεση(*) ή αφαίρεση(/). Οι προσωρινές μεταβλητές που δημιουργούνται από τις expression και term χρησιμοποιούνται και πάλι από τις ίδιες για να συνεχιστεί επαναληπτικά η διαδικασία μετάφρασης της έκφρασεις σε ενδιάμεση γλώσσα. Το τελικό αποτέλεσμα ανατίθεται σε μια προσωρινή μεταβλητή την οποία επιστρέφει η expression και έπειτα αν π.χ. η έκφραση βρίσκεται στο δεξιό μέλος μιας ανάθεσης η προσωρινή μεταβλητή που επιστράφηκε ανατίθεται εκ νέου στην πραγματική μεταβλητή στην οποία θα γινόταν η ανάθεση.

5.3 Συνθήκες

Οι συνθήκες εξετάζονται αρχικά από την συνάρτηση της γραμματικής **condition** η οποία είναι και υπεύθυνη για την δημιουργία του κώδικα σε ενδιάμεση γλώσσα. Αρχικά η συνάρτηση **boolfactor** εξετάζει τρείς περιπτώσεις, πρώτον εάν υπάρχει **not** και ακουλουθεί κάποιο condition, δεύτερον αν υπάρχει condition μέσα σε αγκύλες([ ]) ή τρίτον εάν έχουμε δυο εκφράσεις και ανάμεσα τους κάποιον σχεσιακό τελεστή. Όποια περίπτωση και εάν ισχύει τελικά όλες καταλήγουν στην τρίτη, δηλαδή με δυο εκφράσεις και ανάμεσα τους κάποιον σχεσιακό τελεστή, οπότε μετά τον κώδικα που θα σχηματιστεί για τις εκφράσεις όπως εξηγήθηκε στην ενότητα Εκφράσεις στο τέλος θα σχηματιστεί μια τετράδα που θα κάνει άλμα αν η συνθήκη ισχύει.Στην περίπτωση που υπάρχει **not** τότε το σχεσιακό σύμβολο αντιστρέφεται μέσω της συνάρτησης **reverse_relop** της ενδιάμεσης γλώσσας. Έπειτα εάν ακουλουθεί **and** μετά από μια συνθήκη τότε η συνάρτηση **boolterm** προσθέτει ένα jump ώστε για την περίπτωση που η συνθήκη είναι ψευδής να γίνει άλμα στο τέλος όλων των συνθηκών. Στο τέλος όλων των συνθηκών υπάρχει ένα jump που κάνει άλμα στο ψευδές κομμάτι του κώδικα. Εάν ακολουθεί **or** τότε η συνάρτηση **condition** δεν προσθέτει κάτι καθώς εάν είναι αληθές το **or** θα γίνει άλμα μέσω της τετράδα που ελέγχει την συνθήκη ενώ αν είναι ψευδές θα συνεχίζει στην επόμενη συνθήκη. Τέλος η συνάρτηση

backpatch της ενδιάμεσης γλώσσας θέτει κατάλληλα όλες τις τετράδες που περιέχουν άλμα υπό συνθήκη ή απλό άλμα στην κατάλληλη τετράδα για άλμα ανάλογα εάν είναι ψευδής ή αληθής ο κώδικας.

5.4 if - else

Για την δημιουργία ορθού κώδικα για το statement **if** αρχικά δημιουργείται ο κώδικας της συνθήκης και μέσω της συνάρτησης **backpatch** της ενδιάμεσης γλώσσας γίνεται εύρεση όλων των αλμάτων υπό συνθήκη τα οποία θέτονται να κάνουν άλμα στο τέλος της συνθήκης όπου ξεκινάνε οι τετράδες για τις οποίες η συνθήκη είναι αληθής. Στο τέλος κάθε συνθήκης όπως αναφέρεται στην ενότητα Συνθήκες υπάρχει ένα άλμα(jump) στο ψευδές κομμάτι κώδικα, έτσι εάν δεν υπάρχει το statement **else** τότε το άλμα αυτό τίθεται μέσω της συνάρτησης backpatch στο τέλος του κώδικα για το αληθές κομμάτι. Αν υπάρχει το statement else τότε στο τέλος του κώδικα για το αληθές κομμάτι προστίθεται ένα άλμα το οποίο οδηγεί στην πρώτη τετράδα μετά το τέλος του ψευδές κομμάτι κώδικα και το άλμα στο τέλος της συνθήκης τίθεται στο κομμάτι κώδικα του else(ψευδές).

5.5 while

Το statement **while** αφού δημιουργήσει την συνθήκη καλεί την συνάρτηση **backpatch** της ενδιάμεσης γλώσσας μέσω της οποίας θέτει όλα τα άλματα υπό συνθήκη στο αληθές κομμάτι του κώδικα και έπειτα θέτει το τελευταίο άλμα της συνθήκης μετά το τέλος του κώδικα που ανήκει στην while. Στο τέλος της while προστίθεται ένα άλμα στην αρχή της ώστε ο κώδικας να τρέχει επαναληπτικά μέχρι η συνθήκη στο while να είναι ψευδής.

5.6 loop

Το statement **loop** προσθέτει στο τέλος του κώδικα που περικλείει ένα άλμα(jump) στην αρχή του κώδικα του. Έτσι οι εντολές του κώδικα του εκτελούνται μέχρι να βρεθεί μια εντολή **exit**. Στο τέλος του κώδικα της loop ελέγχονται όλες οι τετράδες και όπου βρεθεί εντολή ενδιάμεσης γλώσσας exit μετασχηματίζεται σε άλμα εκτός του κώδικα της loop.

5.7 forcase

Στο statement **forcase** για κάθε **when** ο κώδικας προσθέτει ένα άλμα(jump) στο τέλος του. Έτσι οποιοδήποτε από τα when είναι αληθές εκτελείτε ο κώδικας του και έπειτα μέσω του άλματος ο κώδικας μεταβένει στην αρχή του forcase. Έαν κανένα when δεν είναι αλήθές τότε ο κώδικας μεταβένει στο **default** που στην ουσία είναι απλός κώδικας μιας και εκτελείτε μια φορά και μετά βγαίνει από την forcase.

5.8 incase

Στο statement **incase** δημιουργούμε αρχικά μια προσωρινή μεταβλητή π.χ. την T_0 και της δίνουμε την τιμή 0. Για κάθε **when** προσθέτουμε στο τέλος του κώδικα του ότι το T_0 = 1. Στο τέλος της incase ελέγχουμε την τιμή του T_0, εάν το T_0 == 0 τότε μεταβαίνουμε εκτός incase μιας και κανένα when δεν εκτελέστηκε, εάν T_0 == 1 τότε τουλάχιστον ένα when εκτελέστηκε και έτσι μεταβαίνουμε με άλμα στην αρχή της incase. Στην αρχή της incase ξαναθέτουμε το T_0 σε 0 και ο βρόχος συνεχίζεται μέχρι να φτάσει στο κομμάτι του ελέγχου της T_0 και να είναι 0, δηλαδή να μην έχει εκτελεστεί ο κώδικας από κανένα when και έτσι να βγούμε εκτός incase.

5.9 doublewhile

Για την ορθή λειτουργεία της **doublewhile** αρχικά δημιουργείται μια προσωρινή μεταβλητή π.χ. η T_0 η οποία τίθεται στο 0 (T_0 = 0). Έπειτα εξετάζεται η συνθήκη και εάν είναι αληθής γίνεται άλμα στο αντίστοιχο κομμάτι κώδικα σε αντίθετη περίπτωση γίνεται άλμα στο ψευδές κομμάτι κώδικα. Έπειτα εκτελείτε ο παρακάτω αλγόριθμος :
Στην αρχή του αληθές κομματιού κώδικα:
1. Αν το T_0 == 2 τότε βγές από την doublewhile
2. statements του αληθές κομματιού κώδικα
3. T_0 = 1
4. Άλμα(jump) στην συνθήκη του doublewhile.

Στην αρχή του ψευδές κομματιού κώδικα:
1. Αν το T_0 == 1 τότε βγές από την doublewhile
2. statements του ψευδές κομματιού κώδικα
3. T_0 = 2

4. Άλμα(jump) στην συνθήκη του doublewhile.
Το T_0 αρχικά είναι 0 έτσι μπαίνει και στα δυο κομμάτια κώδικα(αληθές ή ψευδές). 'Επειτα εάν το doublewhile μπεί μια φορά στο αληθές κομμάτι κώδικα το T_0 γίνεται 1 (βήμα 3 - αληθές) έτσι αν η συνθήκη κάποια στιγμή γίνει ψευδής ο κώδικας θα αναγκαστεί να βγεί απο το doublewhile καθώς το T_0 == 1(βήμα 1 - ψευδές). Παρόμοια εάν πρώτα μπεί στο ψευδές κομμάτι το T_0 θα γίνει 2 (βήμα 3 - ψευδές) όταν κάποια στιγμή η συνθήκη γίνει αληθής θα γίνει έξοδος από την doublewhile(βήμα 1 - αληθές).

Κεφάλαιο 6

Πίνακας Συμβόλων

Ο πίνακας συμβόλων αποτελείται από δυο κλάσεις την **array_of_symbols** και την **function_activity_record**. Η **function_activity_record** χρησιμοποιείται στην ουσία σαν μια δομή δεδομένων έχωντας μέσα της τα πεδία που χρειάζεται για να γίνουν έλεγχοι. Τα πεδία αυτά είναι το όνομα της συνάρτησης(**name**), ο τύπος της function ή procedure(**type**), ο αριθμός της τετράδας που ξεκινάει η εν λόγω συνάρτηση(l**starting_quad**), τα ορίσματα της αν έχει (**arguments**) π.χ. arguments = [['in','x'],['inout','y']], οι τοπικές μεταβλητές της που δηλώθηκαν με declare(**variables**), ο αριθμός των προσωρινών μεταβλητών (**temporary_variables**) π.χ αν temporary_variables = 3 τότε στον ενδιάμεσο κώδικα αυτή της συνάρτησης έχουν χρησιμοποιηθεί οι προσωρινές μεταβλητές T_0,T_1,T_2, το επίπεδο φωλιάσματος (**nesting_level**) και το μήκος του εγγραφήματος δραστηροποίησης(**frame_length**) που αθροίζει τα arguments,variables και temporary variables για να βρει πόσο χώρο θα χρειαστεί αργότερα στην στοίβα η συνάρτηση.
Η κλάσση **array_of_symbols** χειρίζεται τον πίνακα συμβόλων αποθηκέυοντας την κάθε συνάρτηση που διαβάζει στην **list_of_functions**. Ο τρόπος που αποθηκεύονται οι συναρτήσεις είναι παρόμοιος με αυτόν της ενδιάμσεσης γλώσσας στην λίστα functions_list, δηλαδή για κάθε νέα συνάρτηση δημιουργείται ένα **function_activity_record** που θα αποθηκεύσει τις πληροφορίες για αυτήν την συνάρτηση και έπειτα οι συναρτήσεις προστίθενται με την σειρά που διαβάζονται, δηλαδή πρώτο είναι πάντα το κύριο πρόγραμμα και ακουλουθούν η συνάρτηση 1ου επιπέδου φωλιάσματος μαζί με τα παιδιά συναρτήσεις και τους απογονούς τους έπειτα η επόμενη συνάρτηση 1ου επιπέδου φωλιάσματος κτλπ. Η συνάρτηση **add_function** προσθέτει μια νέα συνάρτηση στην λίστα συναρτήσεων list_of_functions εάν δεν υπάρχει άλλη στο ίδιο επίπεδο φωλιάσματος με το ίδιο όνομα, τον ίδιο τύπο και τα ίδια ορίσματα,

εάν αυτή η συνάρτηση έχει ξαναδηλωθεί επιστρέφει False και ο mlc εμφανίζει το ανάλογο μήνυμα λάθους αλλιώς συνεχίζει την μετάφραση.. Η συνάρτηση **add_variable** προστέτει μια νέα μεταβλητή που έχει δηλωθεί με declare στην λίστα variables του function_activity_record της τρέχουσας συνάρτησης, εάν αυτή η μεταβλητή έχει ξαναδηλωθεί επιστρέφει False και ο mlc εμφανίζει το ανάλογο μήνυμα λάθους αλλιώς συνεχίζει την μετάφραση. Η συνάρτηση **add_temporary_argument** προσθέτει προσωρινά τα ορίσματα στην λίστα **temporary_arguments** και όταν τα ορίσματα τελειώσουν η συνάρτηση **get_temporary_arguments** επιστρέφει τα ορίσματα που αποθηκεύτηκαν προσωρινά στην λίστα temporary_arguments ώστε να χρησιμοποιηθούν με την συνάρτηση add_function η οποία θα προσπαθήσει να εισάγει την νέα συνάρτηση. Η συνάρτηση **undo_nesting_level** χρησιμοποιείται για να πάμε ένα βάθος φωλιάσματος πίσω όταν τελειώνουμε την μετάφραση μιας συνάρτησης παιδιού και επιστρέφουμε στην γονική της συνάρτηση. Η συνάρτηση **set_temp_variables** θέτει τον αριθμό των προσωρινών μεταβλητών της μορφής T_x οπού x ένας αριθμός. Η συνάρτηση **set_starting_quad** θέτει τον αριθμό της αρχικής τετράδας της συνάρτησης η οποία παράγεται μετά την εγγραφή των τετράδων στο αρχείο ενδιάμεσης γλώσσας από την συνάρτηση write_list του αντικειμένου inLan(για την ενδιάμεση γλώσσα). Η συνάρτηση **current_function_name** επιστρέφει το όνομα της τρέχουσας συνάρτησης και χρησιμοποιήται κυρίως για την εμφάνιση του ονόματος της σε κάποιο μήνυμα λάθους. Η συνάρτηση **undeclared_variable** ελέγχει τις μεταβλητές οι οποίες χρησιμοποιούνται μέσα στον κώδικα μιας συνάρτησης, εάν η μεταβλήτη που χρησιμοποιήται ανήκει στα ορίσματα ή στις τοπικές μεταβλητές της συνάρτησης(που δηλώθηκαν με declare) ή ανήκει στα ορίσματα ή τις τοπικές μεταβλητές μιας γονικής συνάρτησης της ή ανήκει στις τοπικές μεταβλητές του κύριου προγράμματος τότε η μεταβλητή αυτή μπορεί να χρησιμοποιηθεί και επιστρέφεται True ώστε να συνεχιστεί η μετάφραση, σε αντίθετη περίπτωση επιστρέφεται False και ο mlc παράγει ένα μήνυμα λάθους. Εάν υπάρχουν παράπανω της μιας μεταβλήτης με το ίδιο όνομα τότε προτεραιότητα έχουν:

1) τοπικές μεταβλητές και ορίσματα της τρέχουσας συνάρτησης
2) τοπικές μεταβλητές και ορίσματα της πιο κοντινής γονικής συνάρτησης
3) τοπικές μεταβλητές του κύριου προγράμματος

Παρόμοια με την undeclared_variable αλλά για συναρτήσεις είναι η συνάρτηση **undeclared_fun_or_proc**, ελέγχει δηλαδή αν μια συνάρτηση που καλείται είναι δηλωμένη. Εάν το κύριο πρόγραμμα καλεί μια συνάρτηση τότε ελέγχονται όλες οι συναρτήσεις στο 1ο βάθος φωλιάσματος

και επιλέγεται αυτή που έχει το ίδιο όνομα,τον ίδιο τύπο και τα ίδια ορίσματα. Εάν μια συνάρτηση καλεί μια άλλη συνάρτηση τότε επιλέγεται αυτή με το ίδιο όνομα,τον ίδιο τύπο και τα ίδια ορίσματα, σε περίπτωση που υπάρχουν πολλαπλές συναρτήσεις με τα ίδια χαρακτηριστικά σε διαφορετικά επίπεδα φωλιάσματος τότε προτεραιότητα έχουν:
1) οι συναρτήσεις παιδία της καλούσας συνάρτησης
2) η ίδια η συνάρτηση εάν καλεί τον ευατό της
3) οι γονικές συναρτήσεις και οι αδερφικές συναρτήσεις με την προυπόθεση ότι οι αδερφικές έχουν οριστεί πιο πριν.
Η συνάρτηση **check_same_args** είναι μια βοηθητική συνάρτηση η οποία ελέγχει τα ορίσματα δυο συναρτήσεων.Η συνάρτηση **calc_framelength** υπολογίζει το μήκος του εγγραφήματος δραστηριοποίησης της συνάρτησης ενώ η συνάρτηση **write_aos** γράφει της πληροφορίες μιας συνάρτησης του πίνακα συμβόλων στο αρχείο του πίνακα συμβόλων.Η συνάρτηση **delete** διαγράφει το αρχείο του πίνακα συμβόλων ενώ η συνάρτηση **close** το κλείνει και το αποθηκεύει.

Κεφάλαιο 7

Errors

Τα Errors απότελειται από τις κλάσεις **bcolors**, **error_types**, **warning_types** και την **error_handler**. Η bcolors περιέχει κάποια πεδία που χρησιμοποιούνται για να δώσουν χρώμα στους χαρακτήρες που εκτυπώνονται με την print. Οι κλάσεις error_types και warning_types είναι απαριθμητές για το είδος του error και warning αντίστοιχα. Η κλάση error_handler περιέχει τις συναρτήσεις **error_handle** και **warning_handle** οι οποίες παίρνουν σαν όρισμα το είδος του error και warning αντίστοιχα(το είδος ανήκει στις error_types και warning_types αντίστοιχα) και ένα μεταβλητό πλήθος ορισμάτων ανάλογα το error-warning. Το πλήθος των ορισμάτων εξετάζεται και σε περίπτωση που δεν υπάρχει κάτι μεμπτό επιστρέφεται True, σε αντίθετη περίπτωση εμφανίζεται μήνυμα λάθους στην περίπτωση του error και γίνεται έξοδος διαγράφοντας όλα τα αρχεία που έχουν δημιουργηθεί ως εκείνη την στιγμή, στην περίπτωση του warning απλά εμφανίζεται ένα μήνυμα προειδοποίησης χωρίς να γίνεται έξοδος. Οι συναρτήσεις **set_inLan**, **set_lex** και **set_aos** χρησιμοποιούνται για να θέσουν τα αντικείμενα που χρησιμοποιούνται από την ενδιάμεση γλώσσα, τον λεκτικό αναλυτή και τον πίνακα συμβόλων αντίστοιχα. Τέλος η **exit_program** όταν κληθεί διαγράφει τα αρχεία που μπορεί να έχουν δημιουργηθεί από την ενδιάμεση γλώσσα και τον πίνακα συμβόλων αντίστοιχα.

Κεφάλαιο 8

Παραγωγή κώδικα assembly MIPS

Η παραγωγή του κώδικα assembly MIPS γίνεται από την κλάση **mips_assembly**. Στον constructor της δημιουργεί το αρχείο .asm για την assembly και ανοίγει το αρχείο .int της ενδιάμεσης γλώσσας, επίσης κρατά των πίνακα συμβόλων για να χρησιμοποιείσει τις πληροφορίες που έχει για τις συναρτήσεις κυρίως στην στοίβα που θα δημιουργήσει για κάθε συνάρτηση. Η συνάρτηση **gnvlcode** αποθηκεύει στον καταχωρητή t0 την διεύθυνση μιας μεταβλητής που δεν ανήκει στην τρέχουσα συνάρτηση αλλά σε κάποια γονική της. Η συνάρτηση **loadvr** μεταφέρει τα δεδομένα μιας μεταβλητής (που είναι αποθηκευμένα στην μνήμη της στοίβας) σε έναν καταχωρητή. Η συνάρτηση **storerv** μεταφέρει τα δεδομένα ενός καταχωρητή σε μια μεταβλητή(που είναι αποθηκευμένη στην μνήμη της στοίβας). Η συνάρτηση **translate_int_to_ass** μεταφράζει τις τετράδες της ενδιάμεσης γλώσσας σε εντολές assembly MIPS με την βοήθεια των υπολοίπων συναρτήσεων της κλάσης. Οι τετράδες αυτές μεταφράζονται απευθείας σε assembly MIPS χωρίς ιδιαίτερη δυσκολία μιας και η ενδιάμεση γλώσσα μοίαζει αρκετά με την assembly MIPS. Για το πέρασμα παραμέτρων σε μια συνάρτηση εντοπίζεται η διεύθυνση της παραμέτρου(εάν δεν είναι κάποιος ακέραιος) και αποθηκεύεται η διεύθυνση της στην στοίβα της νέας συνάρτησης. Η συνάρτηση **add_command** προσθέτει μια νέα εντολή assembly MIPS στο αρχείο της assembly. Η συνάρτηση **find_function_pos_sq** βρίσκει την θέση μιας συνάρτησης στον πίνακα συμβόλων βάση της αρχικής της τετράδας. Η συνάρτηση **represents_int** ελέγχει εάν ένα αλφαριθμητικό είναι ακέραιος ή οχι. Η συνάρτηση **find_function_pos** βρίσκει και επιστρέφει την θέση μιας συνάρτησης στον πίνακα συμβόλων βάση του ονόματος,του τύπου και των ορισματών της(αν έχει) τα οποία εχουν δηλωθεί στην ενδιάμεση

γλώσσα, π.χ. par...,par,...,call. Η συνάρτηση **check_same_args** είναι μια βοηθητική συνάρτηση η οποία ελέγχει τα ορίσματα δυο συναρτήσεων. Τέλος η συνάρτηση **find_variable_in_parent** βρίσκει μια μεταβλητή σε κάποια από της γονικές συναρτήσεις είτε αυτή είναι τοπική, είτε προσωρινή, είτε κάποιου είδους ορίσματος και επιστρέφει την απόσταση της στην στοίβα της συνάρτησης που ανήκει.

Κεφάλαιο 9

Δημιουργία κώδικα προσομοίωσης C

Η παραγωγή του κώδικα σε C γίνεται από την κλάση **create_c_code**. Στον constructor της δημιουργεί το αρχείο .c για την C και ανοίγει το αρχείο .int της ενδιάμεσης γλώσσας, επίσης κρατά των πίνακα συμβόλων για να χρησιμοποιείσει τις πληροφορίες που έχει για τις συναρτήσεις.Να σημειωθεί ότι ο κώδικας σε C είναι μόνο για testing και για τίποτα άλλο, δηλαδή για να δούμε εάν ο κώδικας σε minimal++ τρέχει αναμενόμενα στην C. Η παραγωγή του κώδικα σε C λειτουργεί μόνο για τις περιπτώσεις που έχουμε το κύριο πρόγραμμα και συναρτήσεις 1ου επιπέδου φωλιάσματος. Εάν έχουμε συναρτήσεις με βάθος φωλιάσματος μεγαλύτερο του 1 τότε η συμπεριφορά του προγράμματος σε C μπορεί να μην είναι η αναμενώμενη. Επίσης οι μεταβλητές που ανήκουν σε κάποια γονική συνάρτηση δεν μπορούν να χρησιμοποιηθούν. Η συνάρτηση **createC** διαβάζει μια προς μια τις γραμμές του αρχείου που περιέχει τον κώδικα σε ενδιάμεση γλώσσα και παράγει τον αντίστοιχο σε C, αξίζει να αναφερθεί μόνο πως τις μεταβλητές που περνάν σαν όρισμα με αναφορά προσομοιώνονται σαν pointer στην C ούτως ώστε σε περίπτωση που αλλαχθεί η τιμή τους, να αλλαχθεί άμεσα και στην συνάρτηση που ανήκουν. Η συνάρτηση **read_line** διαβάζει την επόμενη γραμμή από το αρχείο της ενδιάμεσης γλώσσας και επιστρέφει την τετράδα. Η συνάρτηση **create_variables** βρίσκει την τρέχουσα συνάρτηση στον πίνακα συμβόλων και παράγει για αυτήν όλες τις μεταβλητές που χρησιμοποιεί τοπικές και προσωρινές. Η συνάρτηση **find_function_pos_sq** βρίσκει μια συνάρτηση στον πίνακα συμβόλων βάση του αριθμού της πρώτης τετράδας της συνάρτησης. Η συνάρτηση **check_inout** ελέγχει εάν κάποιο από τα ορίσματα που παίρνει η συνάρτηση περνιέται με αναφορά και αν περνιέται τότε στην συνάρτηση που θα δημιουργηθεί σε C το όρισμα

παίρνει το σύμβολο (’ *)’ του pointer σε αντίθετη περίπτωση δεν παίρνει κάτι ώστε να περαστεί με τιμή.