ΜΕΤΑΦΡΑΣΤΕΣ (ΜΥΥ802)

ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ 2018/2019

Γλώσσα : Starlet

Ονοματεπώνυμο : ΣΑΒΒΟΠΟΥΛΟΣ ΠΕΤΡΟΣ

ΑΜ : 2530

Username : cse32530

Ονοματεπώνυμο : ΒΑΣΙΛΕΙΟΥ ΙΩΑΝΝΗΣ

ΑΜ : 2647

Username : cse42647

Περιεχόμενα

1. ΕΙΣΑΓΩΓΗ

Ένας μεταφραστής αποτελείται από λεκτικό αναλυτή, συντακτικό αναλυτή , ενδιάμεσο κώδικα και τελικό κώδικα.

Το πρόγραμμά υλοποιεί σε γλώσσα python ένα μέρος μεταφραστή για την γλώσσα Starlet.

Με την άσκηση αυτή μπορεί κάποιος να καταλάβει τις βασικές λειτουργίες ενός μεταφραστή και πως περνάνε τα πρώτα στάδια της μετάφρασης.

Ο κώδικάς αποτελείτε από 2 βασικά σημεία, τον Λεκτικό Αναλυτή και τον Συντακτικό Αναλυτή.

2. ΛΕΚΤΙΚΟΣ ΑΝΑΛΥΤΗΣ

Το αλφάβητο της Starlet 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 («(»,«)»,«[»,«]»)
- και διαχωρισμού σχολίων (/* , */ , //)

Παρατηρήσεις στις Λεκτικές Μονάδες :

1. Τα σύμβολα «[» και «]» χρησιμοποιούνται στις λογικές παραστάσεις.
2. Τα σύμβολα «(» και «)» χρησιμοποιούνται στις αριθμητικές παραστάσεις.

Η γλώσσα έχει επίσης ένα σύνολο Δεσμευμένων Λέξεων :

program, endprogram
declare
if , then , else , endif
while , endwhile , dowhile , enddowhile
loop , endloop , exit
forcase , endforcase , incase , endincase , when , default , enddefault
function , endfunction , return , in , inout , inandout
and , or , not
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 πρώτα γράμματα. 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 ή 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 πρώτα κλείσουν. Δεν υποστηρίζονται εμφωλευμένα δηλαδή σχόλια.

Τύποι και δηλώσεις μεταβλητών :

Ο μοναδικός τύπος δεδομένων που υποστηρίζει η Starlet 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 των μεταβλητών αναγνωρίζεται με το ελληνικοί ερωτηματικό. Επιτρέπεται να έχουμε περισσότερες των μία συνεχόμενες χρήσεις της **declare**.

Τελεστές και εκφραστές:

Η προτεραιότητα των τελεστών από την μεγαλύτερη στην μικρότερη είναι :

1. Μοναδιαίοι λογικοί: «not»
2. Πολλαπλασιαστικοί: «*», «/»
3. Μοναδιαίοι προσθετικοί: «+», «-»
4. Δυαδικοί προσθετικοί: «+», «-»
5. Σχεσιακοί «=», «<», «>», «<>», «<=», «>=»
6. Λογικό «and»
7. Λογικό «or»

Αρχικά για τον λεκτικό αναλυτή, δημιουργήθηκε η συνάρτηση **lex** η οποία μας επιστρέφει 2 πράγματα, την επόμενη λεκτική μονάδα και ένα κωδικό που περιγράφει το είδος της λέξης. Για την υλοποίηση αυτού, χρησιμοποιούνται οι global μεταβλητές **‘token’** και **‘tk’**.

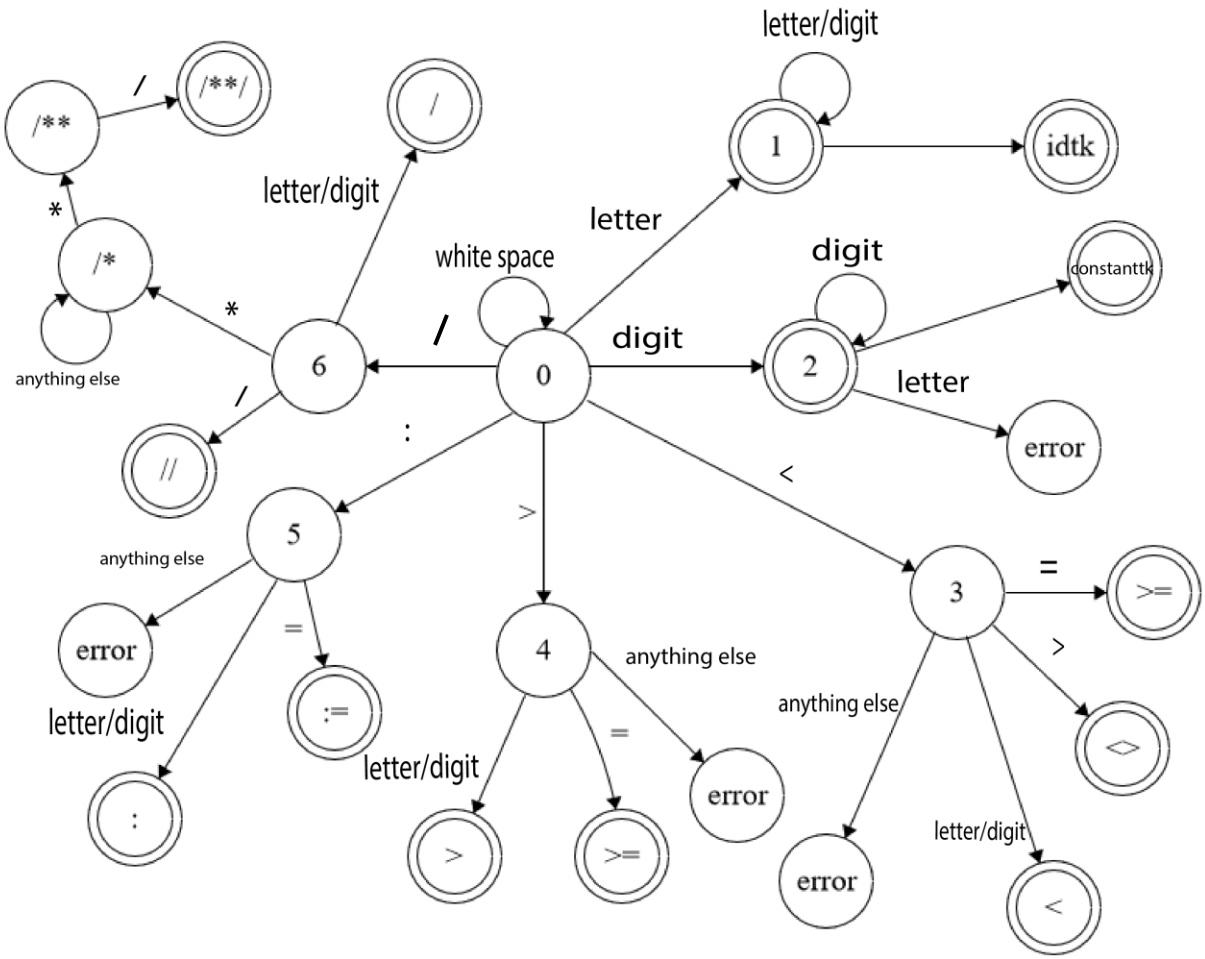
```
keywords = {
                'program' : 'programtk',
                'endprogram' :  'endprogramtk',
                'declare' : 'declaretk',
                'if' : 'iftk',
                'then' : 'thentk',
                'else' : 'elsetk',
                'endif' : 'endiftk',
                'while' : 'whiletk',
                'endwhile' : 'endwhiletk',
                'dowhile' : 'dowhiletk',
                'enddowhile' : 'enddowhiletk',
                'loop' : 'looptk',
                'endloop' : 'endlooptk',
                'exit' : 'exittk',
                'forcase' : 'forcasetk',
                'endforcase' : 'endforcasetk',
                'incase' : 'incasetk',
                'endincase' : 'endincasetk',
                'when' : 'whentk',
                'default' : 'defaulttk',
                'enddefault' : 'enddefaulttk',
                'function': 'functiontk',
                'endfunction' : 'endfunctiontk',
                'return' : 'returntk',
                'in' : 'intk',
                'inout' : 'inouttk',
                'inandout' : 'inandouttk',
                'and' : 'andtk',
                'or' : 'ortk',
                'not' : 'nottk',
                'input' : 'inputtk',
                'print' : 'printtk'             }
```

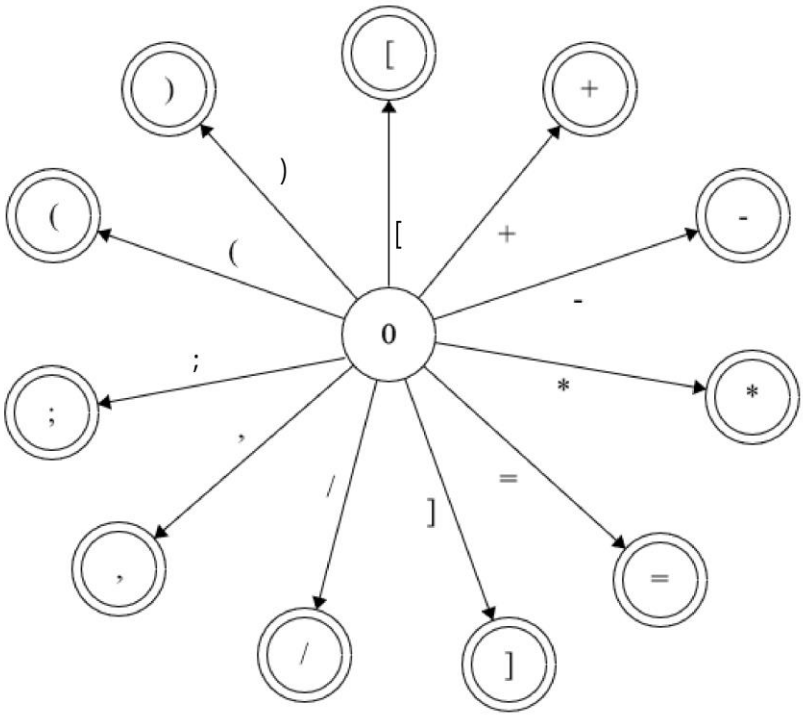
Η συνάρτηση **lex()** διαβάζει χαρακτήρα – χαρακτήρα (στην μεταβλητή **'ch'**) από το αρχείο μέχρι να μην υπάρχει άλλος χαρακτήρας (μέχρι να βρεθεί EOF). Η Κύρια δουλειά της συνάρτησης είναι να ελέγχει εάν ανήκει στην γλώσσα ο χαρακτήρας που έχουμε κάθε φορά στην είσοδο.

Οι λευκοί χαρακτήρες πρέπει να αγνοούνται και αυτό υλοποιείται με μία while.

```
while(ch.isspace()):
    if(ch == '\n'):
        line += 1
    ch = f.read(1)
    positionOfChar = f.tell()
```

Εφόσον **<u>δεν</u>** είναι κάποιος άκυρος χαρακτήρας , αποθηκεύεται σε ένα String (μεταβλητή **'word'**) το οποίο στην αρχή είναι κενό και γεμίζει χαρακτήρα – χαρακτήρα με κάθε σωστό διάβασμα.


letter/digit
/
/**
/**/
/
letter/digit
1
idtk
letter
*
/*
white space
digit
constanttk
*
/
digit
anything else
6
0
2
letter
/
:
error
//
<
5
>
3
=
>=
anything else
error
=
4
anything else
>
letter/digit
:=
letter/digit
=
anything else
:
>
>=
error
error
letter/digit
<


[
)
+
)
(
[
+
-
(
0
-
;
*
*
;
,
/
=
]
=
,
/
]

Η επόμενη δουλειά είναι η αναγνώριση και ταυτοποίηση των χαρακτήρων σε λέξη.

Ο λεκτικός αναλυτής εσωτερικά λειτουργεί σαν ένα **αυτόματο καταστάσεων** το οποίο ξεκινά από μια αρχική κατάσταση, με την είσοδο κάθε χαρακτήρα αλλάζει κατάσταση έως ότου συναντήσει μια τελική κατάσταση. Για την αναγνώριση χρησιμοποιείται το παραπάνω αυτόματο.

Υπάρχουν 3 κατηγορίες χαρακτήρων στην Starlet. Γράμμα, αριθμός και σύμβολο.
Για τα σύμβολα υπάρχει η λίστα symbols που περιέχει όλα τα σύμβολα που δέχεται η γλώσσα.

```
symbols = ['+', '-', '*', '/', '<', '>', '=', '<=', '>=', '<>', ':=' , ';', ',', ':', '(', ')', '[', ']' , '/*', '*/', '//' ]
```

Για τα ψηφία ή γράμματα , υπάρχουν έτοιμες συναρτήσεις στην python που εξυπηρετούν τον σκοπό αυτό. Στην άσκηση χρησιμοποιήθηκαν οι :
isdigit() -> ελέγχει αν ένας χαρακτήρας είναι αριθμός , και αν είναι επιστρέφει true
isalpha() -> ελέγχει αν ένας χαρακ΄τηρας είναι γράμμα, και αν είναι επιστρέφει true.

Όταν το αυτόματο βρεθεί σε αναγνωριστικό η λέξη μας έχει ολοκληρωθεί, την φορτώνουμε στην μεταβλητή ‘’token’’ και αδειάζουμε το String στο οποίο αποθηκεύαμε τους χαρακτήρες.

Πολύ σημαντικό να μην χάσουμε τον τελευταίο χαρακτήρα που μόλις διαβάσαμε και βρεθήκαμε σε αναγνωριστικό.

```
f.seek(positionOfChar-1)
```

3. ΓΡΑΜΜΑΤΙΚΗ

<program> ::= **program** id <block> **endprogram**

<block> ::= <declarations> <subprograms> <statements>

<declarations> ::= (**declare** <varlist> ;)*

<varlist> ::= ε | id (, id)*

<subprograms> ::= (<subprogram>) *

<subprogram> ::= **function** id <funcbody> **endfunction**

<funcbody> ::= <formalpars> <block>

<formalpars> ::= **(** <formalparlist> **)**

<formalparlist> ::= <formalparitem> (, <formalparitem>)* | ε

<formalparitem> ::= **in** id | **inout** id | **inandout** id

<statements> : := <statement> (; <statement>)*

<statement> ::= ε |

<assignment-stat> |

<if-stat> |

<while-stat> |

<do-while-stat> |

<loop-stat> |

<exit-stat> |

<forcase-stat> |

<incase-stat> |

<return-stat> |

<input-stat> |

<print-stat>

<assignment-stat> ::= id **:=** <expression>

<if-stat> ::= **if (**<condition>**) then** <statements> <elsepart> **endif**

<elsepart> ::= ε | **else** <statements>

<while-stat> ::= **while (** <condition> **)** <statements> **endwhile**

<do-while-stat> ::= **dowhile** <statements> **enddowhile (** <condition> **)**

<loop-stat> ::= **loop** <statements> **endloop**

<exit-stat> ::= **exit**

<forcase-stat> ::= **forcase**

(**when (** <condition> **)** : <statements>)*

default : <statements> **enddefault**

endforcase

<incase-stat> ::= **incase**

(**when** **(** <condition> **)** : <statements)*

endincase

<return-stat> ::= **return** <expression>

<print-stat> ::= **print** <expression>

<input-stat> ::= **input** id

<actualpars> ::= **(** <actualparlist> **)**

<actualparlist> ::= <actualparitem> (, <actualparitem>)* | ε

<actualparitem> ::= **in** <expression> | **inout** id | **inandout** id

<condition> ::= <boolterm> (**or** <boolterm>)*

<boolterm> ::= <boolfactor> (**and** <boolfactor>)*

<boolfactor> ::= **not** **[** <condition> **]** | **[** <condition> **]** |

<expression> <relational-oper> <expression>

<expression> ::= <optional-sign> <term> (<add-oper> <term>)*

<term> ::= <factor> (<mul-oper> <factor>)*

<factor> ::= constant | **(** <expression> **)** | id <idtail>

<idtail> ::= ε | <actualpars>

<relational-oper> ::= **=** | **<=** | **>=** | **>** | **<** | **<>**

<add-oper> ::= **+** | **-**

<mul-oper> ::= ***** | **/**

<optional-sign> ::= ε | <add-oper>

4. ΔΟΜΕΣ ΤΗΣ ΓΛΩΣΣΑΣ

Εκχώρηση

Id := expression

Χρησιμοποιείται για την ανάθεση της τιμής μίας μεταβλητής ή μίας σταθεράς, ή μίας έκφρασης σε μία μεταβλητή.

Απόφαση if

if** condition **then
    statements
*[**else***
    statements]
endif

Η εντολή απόφασης **if** εκτιμάει εάν ισχύει η συνθήκη condition και εάν πράγματι ισχύει, τότε εκτελούνται οι εντολές που ακολουθούν το **then** έως ότου συναντηθεί **else** ή **endif**. Το else δεν αποτελεί υποχρεωτικό τμήμα της εντολής και γι' αυτό βρίσκεται σε αγκύλη. Οι εντολές που το ακολουθούν εκτελούνται εάν η συνθήκη condition δεν ισχύει. Το **endif** είναι υποχρεωτικό τμήμα της εντολής.

Επανάληψη while

***while** (condition)*
    (statements)
endwhile

Η εντολή επανάληψης **while** επαναλαμβάνει συνεχώς τις εντολές statements που βρίσκονται ανάμεσα στο **while** και στο **endwhile** , 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.

Επανάληψη dowhile – enddowhile

dowhile

(statements)

enddowhile *(condition)*

Η εντολή επανάληψης **dowhile - enddowhile** επαναλαμβάνει συνεχώς τις εντολές statements που βρίσκονται ανάμεσα στο **dowhile** και στο **endwhile** , όσο η συνθήκη condition ισχύει. Οι statements εκτελούνται τουλάχιστον μια φορά, πριν αποτιμηθεί η condition.

Επανάληψης loop

loop

(statements)

endloop

Η εντολή επανάληψης **loop** επαναλαμβάνει για πάντα τις εντολές statements που βρίσκονται ανάμεσα στο **loop** και στο **endloop** . Έξοδος από τον βρόχο γίνεται όταν κληθεί η εντολή **exit**.

Επανάληψη forcase

forcase

(**when** (condition) : statements)*

default : statements **enddefault**

endforcase

Η δομή επανάληψης **forcase** ελέγχει τις condition που βρίσκονται μετά τα **when**. Μόλις μια από αυτές βρεθεί αληθής, τότε εκτελούνται οι statements που ακολουθούν. Μετά ο έλεγχος μεταβαίνει έξω από την **forcase**. Αν καμία από τις **when** δεν ισχύει , τότε ο έλεγχος μεταβαίνει στη **default** και εκτελούνται οι αντίστοιχες statements. Στη συνέχει ο έλεγχος μεταβαίνει στην αρχή της **forcase**.

Επανάληψη incase

incase
(**when** (condition) : statements)*
endincase

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 α 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, εάν έσω και μία από τις statements έχει εκτελεστεί.

Επιστροφής τιμής

return *expression*

Χρησιμοποιείται μέσα σε συναρτήσεις για να επιστραφεί το αποτέλεσμά της.

Εξόδος

print *expression*

Εμφανίζει στην οθόνη το αποτέλεσμα της αποτίμησης του *expression.*

Εισόδος

input *id*

Ζητάει από τον χρήστη να δώσει μία τιμή μέσα από το πληκτρολόγιο.

5. ΥΠΟΠΡΟΓΡΑΜΜΑΤΑ

Η Starlet υποστηρίζει συναρτήσεις.

function *id (formal_pars)*
declarations
subprograms
statements
endfunction

Η «formal_pars» είναι η λίστα των τυπικών παραμέτρων. Οι συναρτήσεις μπορούν 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.

Μετάδοση Παραμέτρων

Η Starlet υποστηρίζει τρείς τρόπους μετάδοσης παραμέτρων.

- Με σταθερή τιμή. Δηλώνεται με τη 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.

- Με αντιγραφή. Δηλώνεται με τη λεκτική μονάδα **inandout.** Κάθε αλλαγή στη τιμή της μεταφέρεται στο πρόγραμμα που κάλεσε τη συνάρτηση, όταν ολοκληρωθεί η εκτέλεση της συνάρτησης.

Στην κλήση μιας συνάρτησης οι πραγματικοί παράμετροι συντάσσονται μετά από τις λέξεις κλειδιά **in**, **inout**, και **inandout**, ανάλογα με το αν περνάνε με τιμή, αναφορά ή αντιγραφή.

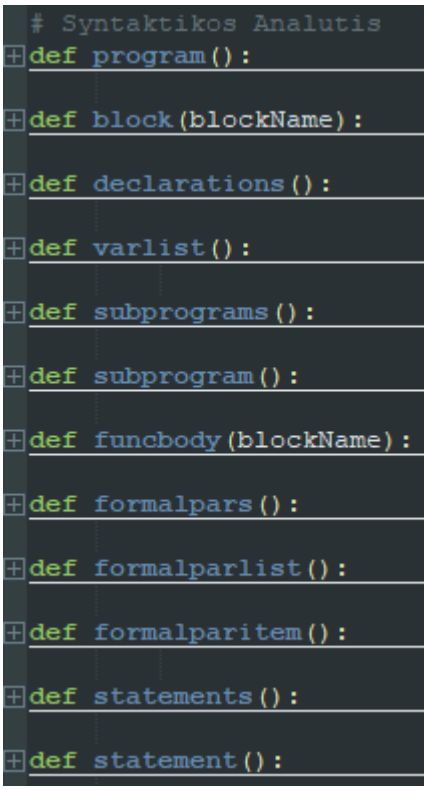
Κατάληξη

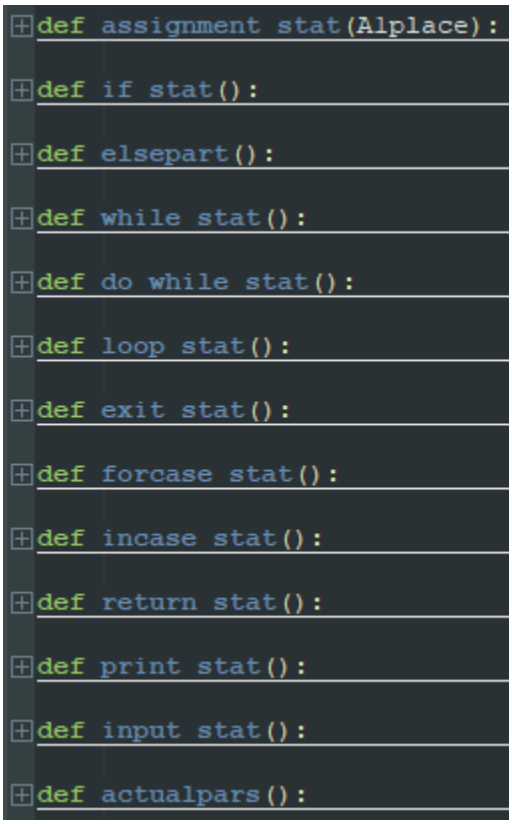
Τα αρχεία της Starlet έχουν κατάληξη .stl

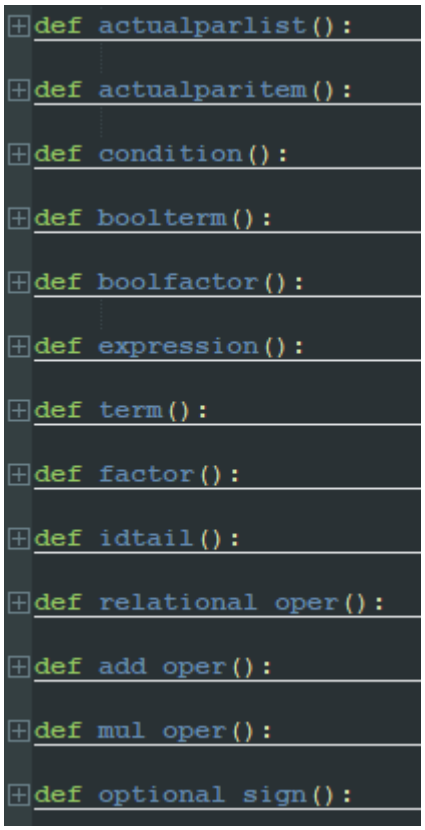
6. ΣΥΝΤΑΚΤΙΚΟΣ ΑΝΑΛΥΤΗΣ

Για την υλοποίηση του Συντακτικού Αναλυτή, η βασική ιδέα είναι να υπάρχει ένα υποπρόγραμμα για κάθε κανόνα της γραμματικής.

Στην περίπτωσή μας , έχει υλοποιηθεί από μια συνάρτηση για κάθε κανόνα της γραμματικής.







Μέσω της συντακτικής ανάλυσης, ο χρήστης ενημερώνεται για το αν έχει συντάξει σωστά το πρόγραμμά του και αυτό ακολουθεί όλους τους κανόνες τις γλώσσας. Καθώς εάν έχει περάσει από τον λεκτικό αναλυτή, δεν θα έχει κάποιο λάθος στις λεκτικές του μονάδες, άρα πρέπει να ελεγχθεί η σύνταξη.

Ο αναλυτής λοιπόν ενημερώνει για τυχόν συντακτικά λάθη, επιστρέφοντας ένα μήνυμα το οποίο περιγράφει το λάθος (συνήθως ποια λεκτική μονάδα έπρεπε να υπάρχει εκεί που βρέθηκε το λάθος) ενημερώνοντας και σε ποια γραμμή είναι το λάθος ώστε να βοηθήσει τον προγραμματιστή στην διόρθωση.

Η παραπάνω λειτουργεία γίνεται με τα παρακάτω βήματα

- Δημιουργία συνάρτησης για κάθε κανόνα
- Εάν βρεθεί μη τερματικό σύμβολο (δηλαδή λεκτική μονάδα) , καλούμε την αντίστοιχη συνάρτηση.
- Όταν βρεθεί τερματικό σύμβολο τότε:
  - Ελέγχουμε αν ο λεκτικός αναλυτής έχει επιστρέψει λεκτική μονάδα η οποία ταυτίζεται με το τερματικό σύμβολο στον γραμματικό κανόνα.
    Αν συμβαίνει αυτό, τότε έχουμε αναγνωρίσει με επιτυχία τη λεκτική μονάδα.
  - Αντίθετα, εάν ο λεκτικός αναλυτής δεν επιστρέψει την λεκτική μονάδα που αναμένει ο συντακτικός αναλυτής, τότε υπάρχει συντακτικό λάθος και ο χρήστης ενημερώνεται με μήνυμα λάθους.
    Το μήνυμα λάθους περιγράφει αναλυτικά σε ποια γραμμή υπάρχει πρόβλημα και ποια λέξη θα έπρεπε να υπάρχει για να είναι συντακτικά σωστός ο κώδικας.
    Στην συνέχεια τερματίζει το πρόγραμμα.

Τέλος , μόλις ολοκληρωθεί η αναγνώριση της τελευταίας λεκτικής μονάδας του πηγαίου προγράμματος, η συντακτική ανάλυση έχει ολοκληρωθεί με επιτυχία.

7. ΕΝΔΙΑΜΕΣΟΣ ΚΩΔΙΚΑΣ

Ο ενδιάμεσος κώδικας είναι ένα σύνολο από τετράδες

- Ένας τελεστής
- Τρία τελούμενα

π.χ. +, a, b, t_1

*, t_1, 2, t_2

:=, t_2, _, c

Οι τετράδες είναι αριθμημένες. Κάθε τετράδα έχει μπροστά της έναν μοναδικό αριθμό που τη χαρακτηρίζει. Μόλις τελειώσει η εκτέλεση μιας τετράδας εκτελείται η τετράδα που έχει τον αμέσως μεγαλύτερο αριθμό, εκτός εάν η τετράδα που μόλις εκτελέστηκε υποδείξει κάτι διαφορετικό.

π.χ. 100: +, a, b, c
110: +, d, e, f

Τετράδες της μορφής : op, x, y, z

- Όπου το op μπορεί να είναι ένα εκ των : +, -, *, /

- Τα τελούμενα x, y μπορεί να είναι :
  - Ονόματα μεταβλητών
  - Αριθμητικές σταθερές

- Το τελούμενο z να είναι :
  - Όνομα μεταβλητής

- Εφαρμόζεται ο τελεστής op στα τελούμενα x και y και το αποτέλεσμα τοποθετείται στο τελούμενο z

π.χ. : +, a, b, c *αντιστοιχεί στην πράξη c=a+b*

/, a, b, c *αντιστοιχεί στην πράξη c=a/b*

Τετράδες της μορφής: :=, x, _, z

- Το τελούμενο x μπορεί να είναι :
  - Όνομα μεταβλητής
  - Αριθμητική σταθερά

- Το τελούμενο z μπορεί να είναι :
  - Όνομα μεταβλητής

- Η τιμή του x εκχωρείται στην μεταβλητή z
  - Αντιστοιχεί στην εκχώρηση z:=x

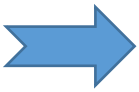
Παράδειγμα.

r := 4

pi := 3.14

area = pi * r * r

θα μπορούσε ισοδύναμα



χωρίς να σημαίνει ότι αυτό ακριβώς θα βγάλει και ο μεταγλωττιστής

100 : :=, 4, _, r

110: :=, 3.14, _, 4

120: *, pi, r, T_1

130: *, T_1, r, area

Τελεστής άλματος χωρίς συνθήκη (jump)

jump, _, _, z μεταπήδηση χωρίς όρους στη θέση z

π.χ. 100: :=, 1, _, x

110: jump 130 *Όταν φτάσουμε στο 130 η τιμή*

120: :=, 2, _, x *του χ θα είναι 1, και όχι 2*

130: ….

Τελεστής άλματος με συνθήκη (relop)

relop, x, y, z μεταπήδηση στην θέση z αν ισχύει η x relop y

όπου relop ένας από τους τελεστές: =, >, <, <>, >=, <=

π.χ. 100: =, a, 4, 120

110: jump, _, _, 140 *Το b θα έχει την τιμή 1, αν ισχύει συνθήκη a=4*

120: :=, 1, _, b *και την τιμή 2, αν δεν ισχύει.*

130: jump 150

140: :=, 2, _, b

150: ….

Αρχή και τέλος ενότητας (begin_block , end_block , name, halt)

π.χ. 100: begin_block, add, _, _

110: :=, 1, _, x

120: :=, 2, _, y

130: +, x, y, z

140: halt, _, _, _

150: end_block, add, _, _

Συναρτήσεις – Διαδικασίες

- **par, x, m, _** όπου x παράμετρος συνάρτησης και m ο τρόπος μετάδοσης.
  CV : μετάδοση με τιμή
  REF: μετάδοση με αναφορά
  RET: επιτροφή τιμής συνάρτησης

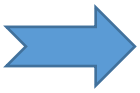
- **call, name, _, _,** κλήση συνάρτησης name

- **ret, x, _, _** επιστροφή τιμής συνάρτησης

***Παράδειγμα* κλήσης συνάρτησης**

X := foo (in a, inout b)

θα μπορούσε ισοδύναμα



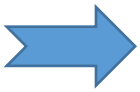
χωρίς να σημαίνει ότι αυτό

ακριβώς θα βγάλει και ο μεταγλωττιστής

100: par, a, CV, _
110 : par, b, REF, _
120: par, T_1, RET, _
130: call, foo, _, _
140: …. Τιμή στο x

Παράδειγμα κλήσης διαδικασίας

call foo (in a, inout b)



χωρίς να σημαίνει ότι αυτό
ακριβώς θα βγάλει και ο μεταγλωττιστής

100: par, a, CV, _
110: par, b, REF, _
120: call, foo, _, _
130: …..

Βοηθητικές Υπορουτίνες

- **nextQuad()**
  Επιστρέφει τον αριθμό της επόμενης τετράδας που πρόκειται να παραχθεί.

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

- **newTemp()**
  Δημιουργεί και επιστρέφει μία νέα προσωρινή μεταβλητή.
  Οι προσωρινές μεταβλητές είναι τις μορφής: T_1, T_2, T_3, ..

- **emptyList()**
  Δημιουργεί μια κενή λίστα ετικετών τετράδων.

- **makeList(x)**
  Δημιουργεί μια λίστα ετικετών τετράδων που περιέχει μόνο το x.

- **mergeList(list1, list2)**
  Δημιουργεί μια λίστα ετικετών τετράδων από τη συνένωση των λιστών list1, list2.

- **backPatch(list, z)**
  Η λίστα list 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.

Η υλοποίηση του ενδιάμεσου κώδικα στην ουσία είναι η σωστή τοποθέτηση των παραπάνω υπορουτίνων στις συναρτήσεις του συντακτικού αναλυτή. Παρακάτω θα γίνει ανάλυση κάθε περίπτωσης. Για την καλύτερη κατανόηση , στα σημεία που πρέπει να μπει κώδικας θα υπάρχουν P1, P2, .. σαν τύπου label και από κάτω θα δίνεται ο κώδικας του κάθε label.

Αριθμητική Παράσταση

$E \rightarrow T^1(+T^2\{P_1\}) * \{P_2\}$

$\{P_1\}$: w = newTemp()

genQuad("+", T^1.place, T^2.place, w)

T^1.place = w

$\{P_2\}$: E.place = T^1.place

Νέα προσωρινή μεταβλητή που θα κρατήσει το μέχρι στιγμής αποτέλεσμα

Παραγωγή τετράδας που προσθέτει το μέχρι στιγμής αποτέλεσμα στο νέο T^2

Το μέχρι στιγμής αποτέλεσμα τοποθετείται στην T^1 ώστε να χρησιμοποιηθεί αν υπάρχει επόμενο T^2

Όταν δεν υπάρχει άλλο T^2 το αποτέλεσμα είναι στο T^1

$T \rightarrow F^1(\text{ x } F^2\{P_1\}) * \{P_2\}$ *Ανάλογη λογική με τον κανόνα E (από πάνω)*

$\{P_1\}$: w = newTemp()

genQuad("+", F^1.place, F^2.place, w)

F^1.place = w

$\{P_2\}$: T.place = F^1.place

$F \rightarrow (\text{ E }) \{P_1\})$

$\{P_1\}$: F.place = E.place Απλή μεταφορά από το E.place στο F.place

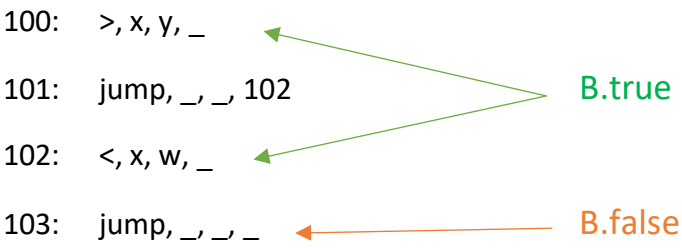
$F \rightarrow \text{id } \{P_1\}$

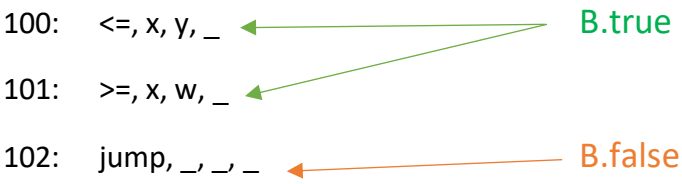
$\{P_1\}$: F.place = id.place Απλή μεταφορά από το id.place στο F.place

Λογικές Παραστάσεις

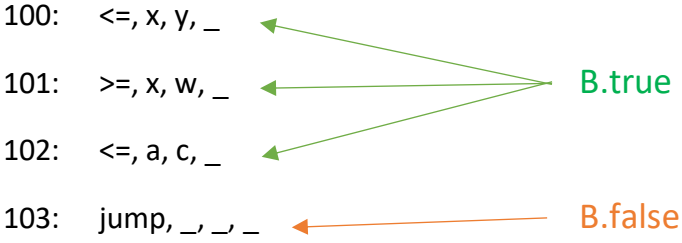
Έστω η γραμματική B → Q (or Q)*

Παράδειγμα **B = x>y or x<w**





Παράδειγμα **B = x>y or x<w or a>c**



$B \rightarrow Q^1 \{P_1\}$ (or $\{P_2\}$ Q^2 $\{P_3\}$)*

$\{P_1\}$: $B.true = Q^1.true$

$B.false = Q^1.false$

$\{P_2\}$: backPatch(B.false, nextQuad())

$\{P_3\}$: $B.true$ = mergeList(B.true, $Q^2.true$)

$B.false = Q^2.false$

Μεταφορά των τετράδων από τη λίστα Q^1 στη λίστα B

Συμπλήρωση όσων τετράδων μπορούν να συμπληρωθούν μέσα στον κανόνα

Συσσώρευση στη λίστα true των τετράδων που δεν μπορούν να συμπληρωθούν και αντιστοιχούν σε αληθή αποτίμηση λογικής παράστασης

Η λίστα false περιέχει την τετράδα η οποία αντιστοιχεί σε στη μη αληθή αποτίμηση της λογικής παράστασης

Έστω η γραμματική Q → R (and R)*

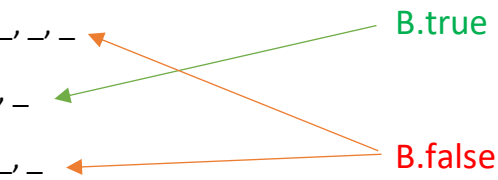
Παράδειγμα B = x>y and x<w

100: >, x, y, 102

101: jump, _, _, _

102: <, x, w, _

103: jump, _, _, _



$Q \rightarrow R^1 \{P_1\}$ (and $\{P_2\}$ R^2 $\{P_3\}$)*

$\{P_1\}$: $Q.true = R^1.true$

$Q.false = R^1.false$

Μεταφορά των τετράδων από τη λίστα R^1 στη λίστα Q

$\{P_2\}$: backPatch(Q.true, nextQuad())

Συμπλήρωση όσων τετράδων μπορούν να συμπληρωθούν μέσα στον κανόνα

$\{P_3\}$: $Q.true = mergeList(Q.false, R^2.false)$

$Q.true = R^2.true$

Συσσώρευση στη λίστα false των τετράδων που δεν μπορούν να συμπληρωθούν και αντιστοιχούν σε μη αληθή αποτίμηση λογικής παράστασης

Η λίστα true περιέχει την τετράδα η οποία αντιστοιχεί σε στην αληθή αποτίμηση της λογικής παράστασης

Έστω η γραμματική R → (B)

$R \rightarrow (B) \{P_1\}$

$\{P_1\}$: R. true = B. true

R. false = B. false

Μεταφορά των τετράδων από τη λίστα B στη λίστα R

Έστω η γραμματική R → E relop E

$R \rightarrow E^1 \text{ relop } E^2 \{P_1\}$

$\{P_1\}$: R. true = makeList(nextQuad())

genQuad(relop, E^1. palce, E^2. place, "_")

R. false = makeList(nextQuad())

genQuad("jump", "_", "_", "_")

Δημιουργία μη συμπληρωμένης τετράδας και εισαγωγή στη λίστα μη συμπληρωμένων τετράδων για την αληθή αποτίμηση της relop

Δημιουργία μη συμπληρωμένης τετράδας και εισαγωγή στη λίστα μη συμπληρωμένων τετράδων για τη μη αληθή αποτίμηση της relop

Εντολή return-stat

$S \rightarrow \text{return } (E) \{P_1\}$

Για το E έχουμε Eplace=expression()

$\{P_1\}$: genQuad("retv", Eplace, "_", "_")

Εκχώρηση

$S \rightarrow id := E \{P_1\}$

$\{P_1\}$: genQuad(" := ", E. place, "_", id)

Δομή while (while-stat)

Για το B χρησιμοποιούμε : whileTrue, whileFalse = condition() (δηλαδή B.true και B.false)

$S \rightarrow \text{while } \{P_1\} \text{ B do } \{P_2\} \text{ } S^2 \text{ } \{P_3\}$

$\{P_1\}$: whileQuad := nextQuad()

$\{P_2\}$: backPatch(whileTrue, nextQuad())

$\{P_3\}$: genQuad("jump", "_" , "_", whileQuad)

backPatch(whileFalse, nextQuad())

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, το true πάνω στην S και το false έξω από τη δομή

Μετάβαση στην αρχή της συνθήκης ώστε να ξαναγίνει έλεγχος

Δομή Repeat...Until (do-while-stat)

$S \rightarrow \text{repeat } \{P_1\} \text{ } S^1 \text{ until (cond) } \{P_2\}$

Για το cond χρησιμοποιούμε : dowhileTrue, dowhileFalse = condition() (δηλαδή cond.true και cond.false)

Οι τετράδες αυτές πρέπει να μεταβούν στην αρχή της συνθήκης για να επανελεγχθεί

$\{P_1\}$: dowholeQuad := nextQuad()

$\{P_2\}$: backPatch(dowhileFalse, dowhileQuad)

backPatch(dowhileTrue, nextQuad())

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, δηλαδή έξω από τη δομή

Δομή if (if-stat)

$S \rightarrow \text{if } B \text{ then } \{P_1\} \; S^1 \; \{P_2\} \; \text{TAIL} \; \{P_3\}$

Για το B χρησιμοποιούμε : ifTrue, ifFalse = condition() (δηλαδή B.true και B.false)

$\{P_1\}$: backPatch (ifTrue, nextQuad())

$\{P_2\}$: ifList = makeList(nextQuad())

genQuad("jump" , "_", "_", "_")

backPatch(ifFalse , nextQuad())

$\{P_3\}$: backPatch (ifList , nextQuad())

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, στο if και else αντίστοιχα

Εξασφαλίζουμε ότι εάν εκτελεστούν οι εντολές του if δε θα εκτελεστούν στη συνέχεια οι εντολές του else

$\text{TAIL} \rightarrow \text{else } S^2 \mid \text{TAIL} \rightarrow \varepsilon$

Είσοδος input-stat

$S \rightarrow \text{input (id)} \; \{P_1\}$

Για το id έχουμε Idplace = word

$\{P_1\}$: genQuad ("inp" , Idplace, "_", "_")

Έξοδος print-stat

$S \rightarrow \text{print (E)} \; \{P_1\}$

Για το E έχουμε Eplace=expression()

$\{P_1\}$: genQuad ("out" , Eplace, "_", "_")

Δομή loop-stat

$S \rightarrow \{P_0\}$ loop S $\{P_1\}$ endloop

$\{P_0\}$: firstQuad = nextQuad()

$\{P_1\}$: backPatch (forLoopList, nextQuad() + 1)

genQuad ("jump" , "_", "_", firstQuad)

Δομή forcase-stat

$S \rightarrow$ forcase $\{P_0\}$

(when (cond) $\{P_1\}$: S^1 $\{P_2\}$)

default : S^2 enddefault $\{P_3\}$

endforcase

Για το cond χρησιμοποιούμε το condTrue, condFalse = condition()

$\{P_0\}$: exitList = emptyList()

forcaseCond = nextQuad()

Κρατάω την επόμενη τετράδα για να ξέρω την αρχή του forcase

$\{P_1\}$: backPatch(condTrue, nextQuad())

$\{P_2\}$: t = makeList(nextQuad())

genQuad = ("jump" , "_", "_", "_")

mergeList(exitList, t)

backPatch(condFalse, nextQuad())

Συμπλήρωση των τετράδων που έχουν μείνει ασυμπλήρωτες και γνωρίζουμε τώρα ότι πρέπει να συμπληρωθούν με την επόμενη τετράδα, στο condition , ή στο επόμενο when

$\{P_3\}$: genQuad = ("jump" , "_", "_", forecaseCond)

backPatch(exitList, nextQuad())

Αφού τελειώσει το default, ξαναπάω στην αρχή του forcase

Δομή incase-stat

S → incase $\{P_0\}$

(when (cond) $\{P_1\}$: S^1 $\{P_2\}$)

endincase $\{P_3\}$

Για το cond χρησιμοποιούμε το condTrue, condFalse = condition()

$\{P_0\}$: flag = newTemp()

firstQuad = nextQuad()

genQuad(" := ", "0", "_", flag)

$\{P_1\}$: backPatch(condTrue, nextQuad())

$\{P_2\}$: genQuad(" := ", "1", "_", flag)

backPatch(condFalse, nextQuad())

$\{P_3\}$: genQuad(" := ", "1", "_", firstQuad)

Αρχή του incase

Κρατάμε σε ένα flag την πληροφορία εάν είναι αλήθεια έστω και 1 when καθώς αν ισχύει πρέπει στο τέλος να ξαναπάμε στην αρχή του incase

Συμπληρώνουμε τις 4άδες που έχουν μείνει ασυμπλήρωτες

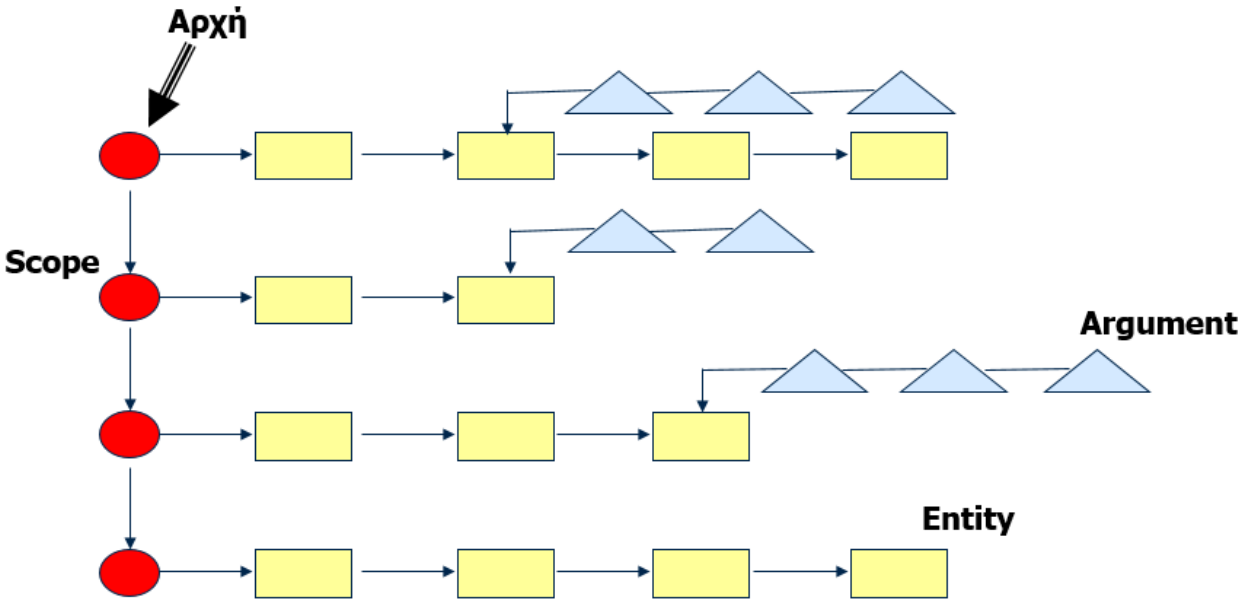
```
# Endiamesos Kwdikas
temp_counter = -1     # counter of T_
quads_line = -1       # counter of line 4adas
var_list = []         # list to store the temp variables ( T_ )
quads = {}            # "line : 4ada"
```

Οι παραπάνω μεταβλητές χρησιμοποιούνται (όπως γράφει στα σχόλια)

- Μετρητής προσωρινών μεταβλητών
- Μετρητής τετράδων
- Λίστα που κρατάει όλες τις προσωρινές μεταβλητές
- Ένα λεξικό, που κρατάει την τετράδα και τον αριθμό γραμμής της ως κλειδί.

8. ΠΙΝΑΚΑΣ ΣΥΜΒΟΛΩΝ

Μορφή του πινακα συμβόλων



Εγγραφές στον Πίνακα

Record Entity

- char[] name
- τύπος
  - μεταβλητή
    - int type
    - int offset (απόσταση από την κορυφή της στοίβας)
  - συνάρτηση
    - int type
    - int startQuad (ετικέτα της πρώτης τετράδας του κώδικα της συνάρτησης)
    - list argument (λίστα παραμέτρων)
    - int framelength (μήκος εγγραφήματος δραστηριοποίησης)
  - σταθερά
    - char[] value (τιμή της σταθεράς)
  - παράμετρος
    - int parMode (τρόπος περάσματος)
    - int offset (απόσταση από την κορυφή της στοίβας)
  - προσωρινή μεταβλητή
    - int offset (απόσταση από την κορυφή της στοίβας)
- pointer Entity next

Record Scope

- pointer Entity (λίστα από Entities)
- int nestingLevel (βάθος φωλιάσματος)
- pointer Scope enclosingScope

Record Argument

- int parMode (τρόπος περάσματος)
- int type (τύπος μεταβλητής)
- pointer Argument next

Εγγράφημα Δραστηριοποίησης

- Δημιουργείται για κάθε συνάρτηση από αυτήν που την καλεί
- Όταν αρχίζει η εκτέλεση της συνάρτησης ο δείκτης στοίβας μεταφέρεται στην αρχή του εγγρασήματος δραστηριοποίησης
- Περιέχει πληροφορίες που χρησιμέυουν για την εκτέλεση και τον τερματισμό της συνάρτησης καθώς και πληροφορίες που σχετίζονται με τις μεταβλητές που χρησιμοποιεί
- Όταν τερματίζεται η συνάρτηση ο χώρος που καταλαμβάνει το εγγράφημα δραστηριοποίησης επιστρέφεται στο σύστημα

- Διεύθυνση επιστροφής: η διεύθυνση στην οποία θα μεταβεί η ροή του προγράμματος όταν ολοκληρωθεί η εκτέλεση της συνάρτησης
- Σύνδεσμος Προσπέλασης: δείχνει στο εγγράφημα δραστηριοποίσης που πρέπει να αναζητηθούν μεταβλητές οι οποίες δεν είναι τοπικές αλλά η συνάρτηση έχει δικαίωμα να χρησιμοποιήσει
- Επιστροφή τιμής: η διεύθυνση στην οποία θα γραφεί το αποτέλεσμα της συνάρτησης όταν αυτό υπολογιστεί

- Χώρος αποθήκευσης παραμέτρων συνάρτησης
  - αποθηκεύεται η τιμή, αν πρόκειται για πέρασμα με τιμή
  - αποθηκεύεται η διεύθυνση, αν πρόκειται για πέρασμα με αναφορά

- Χώρος αποθήκευσης τοπικών μεταβλητών

- Χώρος αποθήκευσης προσωρινών μεταβλητών

Ενέργειες στον πίνακα συμβόλων

- Προσθήκη νέου Scope: όταν ξεκινάμε τη μετάφραση μιας νέας συνάρτησης
- Διαφραφή Scope: όταν τελειώνουμε τη μετάφραση μιας συνάρτησης - με τη διαγραφή διαγράφουμε την εγγραφή (record) του Scope και όλες τις λίστες με τα Entity και τα Argument που εξαρτώνται από αυτήν
- Προσθήκη νέου Entity
  - όταν συναντάμε δήλωση μεταβλητής
  - όταν δημιουργείται νέα προσωρινή μεταβλητή
  - όταν συναντάμε δήλωση νέας συνάρτησης
  - όταν συναντάμε δήλωση τυπικής παραμέτρου συνάρτησης
- Προσθήκη νέου Argument: όταν συναντάμε δήλωση τυπικής παραμέτρου συνάρτησης
- Αναζήτηση: μπορεί να αναζητηθεί κάποιο entity με βάση το όνομά του.

  Η αναζήτηση ενός entity γίνεται ξεκινώντας από την αρχή του πίνακα και την πρώτη του γραμμή. Αν δε βρεθεί πηγαίνουμε στην επόμενη γραμμή έως ότου βρεθεί το entity ή τελειώσουν όλα τα entities οπότε επιστρέφουμε και μήνυμα λάθους. Αν με το ζητούμενο όνομα υπάρχει πάνω από ένα entity τότε επιστρέφουμε το πρώτο που θα συναντήσουμε

Για την άσκηση έχουν δημιουργηθεί 2 βοηθητικές συναρτήσεις

- *record_entity(charName, type, argList)*
- *record_arguments(name, type)*

```
# Pinakas Sumvolwn
mainFrameLength=0
recordScopeList = {} # "scopeNestingLevel : entitites"
recordEntityList = []
recordArgumentsList = []
offsetTable = [] # Store all offsets for each nestingLevel  (Scopes)
offsetTable.append(12) # first offset of main
nestingLevel = 0
```

Οι παραπάνω μεταβλητές – λίστες χρησιμοποιήθηκαν για την υλοποίηση του κώδικα για τον πίνακα συμβόλων. Αναλυτικά με την σειρά :

- το μήκος της main που έπρεπε να το αποθηκεύσουμε ¨κάπου¨ καθώς το θέλουμε στο τέλος
- ένα λεξικό στο οποίο κρατάει όλα τα βάθη φωλιάσματος μαζί με τις εγγραφές
- μια λίστα με όλες τις εγγραφές
- μια λίστα με όλα τα ορίσματα
- μια λίστα με τα offset για κάθε βάθος φωλιάσματος
- γέμισμα με 12 (που χρειάζεται κάθε κλήση-συνάρτηση στην αρχή)
- βάθος φωλιάσματος

9. ΣΗΜΑΣΙΟΛΟΓΙΚΗ ΑΝΑΛΥΣΗ

Στην σημασιολογική ανάλυση έχουν συμπεριληφθεί τα εξής :

- Κάθε συνάρτηση έχει μέσα της τουλάχιστον ένα return
- Δεν υπάρχει return έξω από συνάρτηση
- Υπάρχει exit μόνο μέσα σε βρόγχους loop – endloop

Επίσης έχουν πραγματοποιηθεί οι εξής απαιτήσεις :

- Κάθε μεταβλητή ή συνάρτηση που έχει δηλωθεί , να μην έχει δηλωθεί πάνω από μία φορά στο βάθος φωλιάσματος στο οποίο βρίσκεται
- Κάθε μεταβλητή, συνάρτηση ή διαδικασία που χρησιμοποιείται, έχει δηλωθεί, και μάλιστα με τον τρόπο που χρησιμοποιείται (σαν μεταβλητή ή σαν συνάρτηση)
- Οι παράμετροι με τις οποίες καλούνται οι συναρτήσεις είναι ακριβώς αυτές με τις οποίες έχουν δηλωθεί και με τη σωστή σειρά

10. ΤΕΛΙΚΟΣ ΚΩΔΙΚΑΣ

Για την παραγωγή τελικού κώδικα γίνονται τα εξης :

- Από κάθε μια εντολή ενδιάμεσου κώδικα παράγουμε τις αντίστοιχες εντολές του τελικού κώδικα

- Κύριες ενέργειες στην φάση αυτή :
  - Οι μεταβλητές απεικονίζονται στην μνήμη (στοίβα)
  - Το πέρασμα παραμέτρων και η κλήση συναρτήσεων

- Θα δημιουργήσουμε κώδικα για τον επεξεργαστή MIPS

Αρχιτεκτονική MIPS

- Καταχωρητές που θα μας φανούν χρήσιμοι :

  - Καταχωρητές προσωρινών τιμών : $t0….$t7
  - Καταχωρητές οι τιμές των οποίων διατηρούνται ανάμεσα σε κλήσεις συναρτήσεων : $s0….$s7
  - Καταχωρητές ορισμάτων : $a0….$a3
  - Καταχωρητές τιμών : $v0, $v1
  - Stack pointer : $sp
  - Frame pointer : $fp
  - Return address $ra

  -

- Εντολές που θα μας φανούν χρήσιμες για αριθμητικές πράξεις :
  - add $t0, $t1, $t2 t0=t1+t2
  - sub $t0, $t1, $t2 t0=t1-t2
  - mul $t0, $t1, $t2 t0=t1*t2
  - div $t0, $t1, $t2 t0=t1/t2

- Εντολές που θα μας φανούν χρήσιμες για μετακίνηση δεδομένων :

➢ move $t0, $t1	t0=t1	μεταφορά ανάμεσα σε καταχωρητές
➢ li $t0, value	t0=value	σταθερά σε καταχωρητή
➢ lw $t1, mem	t1=[mem]	περιεχόμενο μνήμης σε καταχωρητή
➢ sw $t1, mem	[mem]=t1	περιεχόμενο καταχωρητή σε μνήμη
➢ lw $t1, ($t0)	t1=[t0]	έμμεση αναφορά σε καταχωρητή
➢ sw $t1, -4($sp)	t1=[$sp-4]	έμμεση αναφορά με βάση τον $sp

- Εντολές που θα μας φανούν χρήσιμες για άλματα :

➢ b label	branch to label
➢ beq $t1, $t2, label	jump to label if $t1=$t2
➢ blt $t1, $t2, label	jump to label if $t1<$t2
➢ bgt $t1, $t2, label	jump to label if $t1>$t2
➢ ble $t1, $t2, label	jump to label if $t1<=$t2
➢ bge $t1, $t2, label	jump to label if $t1>=$t2
➢ bne $t1, $t2, label	jump to label if $t1<>$t2

- Εντολές που θα μας φανούν χρήσιμες στην κλήση συναρτήσεων :

➢ j label	jump to label
➢ jal label	κλήση συνάρτησης
➢ jr $ra	άλμα στη διεύθυνση που έχει ο καταχωρητής, στο παράδειγμα είναι ο $ra που έχει την διεύθυνση επιστροφής τιμής

Βοηθητικές Συναρτήσεις

```
# Usefull Functions For Final Code
def gnlvcode(v):

def searchvar(var):

def loadvr(v, r):

def storerv(r, v):

def checkParameters(counterOfPar, numberOfQuad):

def create FinalCode(counterOfQuads):
```

gnvlcode

- μεταφέρει στον $t0 την διεύθυνση μια **μη τοπικής** μεταβλητής
- από τον πίνακα συμβόλων βρίσκει πόσα επίπεδα επάνω βρίσκεται η μη τοπική μεταβλητή και μέσα από τον σύνδεσμο προσπέλασης την εντοπίζει

```
lw  $t0, -4($sp)          στοίβα του γονέα

όσες φορές χρειαστεί :
lw  $t-, -4($sp)          στοίβα του προγόνου που έχει τη μεταβλητή

add  $t0, $t0, -offset    διεύθυνση της μη τοπικής μεταβλητής
```

searchvar

- Βρίσκει το βάθος φωλιάσματος του “var” (nesting level)

checkParameters

- Βρίσκει αριθμό του scope
- Βρίσκει τον αριθμό των ορισμάτων
- Ελέγχει αν είναι σωστός ο αριθμός των ορισμάτων και παραμέτρων των συναρτήσεων

loadvr

- μεταφορά δεδομένων στον καταχωρητή r
- η μεταφορά μπορεί να γίνει από τη μνήμη (στοίβα)
- ή να εκχωρηθεί στο r μια σταθερά
- η σύνταξη της είναι loadvr (v, r)
- διακρίνουμε περιπτώσεις :
  - αν v είναι σταθερά
    li $tr, v
  - αν v είναι καθολική μεταβλητή – δηλαδή ανήκει στο κυρίως πρόγραμμα
    lw $tr, -offset ($s0)
  - αν v είναι τοπική μεταβλητή ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος ίσο με το τρέχον ή προσωρινή
    lw $tr, -offset ($sp)
  - αν v είναι τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος ίσο με το τρέχον
    lw $t0, -offset ($sp)
    lw $tr, ($t0)
  - αν v είναι τοπική μεταβλητή ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος μικρότερο από το τρέχον
    gnlvcode()
    lw $tr, ($t0)
  - αν v είναι τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος μικρότερο από με το τρέχον
    gnlvcode()
    lw $t0, ($t0)
    lw $tr, ($t0)

storerv

- μεταφορά δεδομένων από τον καταχωρητή r στη μνήμη (μεταβλητή v)
- η σύνταξη της είναι storevr(r, v)
- διακρίνουμε με περιπτώσεις :
  - αν v είναι καθολική μεταβλητή – δηλαδή ανήκει στο κυρίως πρόγραμμα
    ```
    sw  $tr, -offset($s0)
    ```
  - αν v είναι τοπική μεταβλητή ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος ίσο με το τρέχον ή προσωρινή
    ```
    sw  $tr, -offset($sp)
    ```
  - αν v είναι τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος ίσο με το τρέχον
    ```
    lw $t0, -offset ($sp)
    lw $tr, ($t0)
    ```
  - αν v είναι τοπική μεταβλητή ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος μικρότερο από το τρέχον
    ```
    gnlvcode()
    lw  $tr, ($t0)
    ```
  - αν v είναι τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος μικρότερο από με το τρέχον
    ```
    gnlvcode()
    lw  $t0, ($t0)
    lw  $tr, ($t0)
    ```

create_FinalCode

Σε αυτήν την συνάρτηση γίνεται σχεδόν όλη η δουλειά. Παράγεται δηλαδή ο τελικός κώδικας.

- Ελέγχει την κάθε τετράδα
- Κάθε τετράδα έχει και τον αριθμό σειρας, έτσι δημιουργούμε όλα τα L_1 , L_2, ..
- Αν είναι η main , τότε δημιουργείται το Lmain

Έτσι με μια σειρά από if καλύπτουμε όλες τις περιπτώσεις, και μετατρέπουμε κάθε λειτουργία του ενδιάμεσου κώδικα , σε τελικό.

Εντολές Αλμάτων

- jump, “_”, “_”, label

  label

- relop(?) , x, y, z

  loadvr(x,1)

  loadvr(y,2)

  branch(?),$t1, $t2, z branch(?) : beq, bne, bgt, blt, bge, ble

Εκχώρηση

- := , x, “_”, z

  loadvr(x,1)

  storerv(1,z)

Εντολές αριθμητικών πράξεων

- op x, y, z

  loadvr(x,1)

  loadvr(y,2)

  op $t1, $t1, $t2 op : add, sub, mul, div

  storerv(1,z)

Εντολές εισόδου - εξόδου

- out “_”, “_”, x

  li $v0, 1

  li $a0, x

  syscall

- in “_”, “_”, x

  li $v0, 5

  syscall

  το αποτέλεσμα γράφεται στον $v0

Επιστροφή τιμής συνάρτησης

- retv “_”, “_”, x

    loadvr(x,1)

    lw $t0, -8($sp)

    sw $t1, ($t0)

    αποθηκεύεται ο x στη διεύθυνση που είναι αποθηκευμένη στην $3^η$ θέση του εγγραφήματος δεαστηριοποίησης

- εναλλακτικά μπορούμε να γράψουμε το αποτέλεσμα στον $v0, και μετά πρέπει να γροντίσουμε να το πάρουμε από εκεί

    loadvr(x,1)

    move $v0, $t1

Παράμετροι Συνάρτησης

- πριν από την πρώτη παράμετρο , τοποθετούμε τον $fp να δείχνει στην στοίβα της συνάρτησης που θα δημιουργηθεί

    add $fp, $sp, framelength

- par, x, CV, _

    loadvr(x,0)

    sw $t0, -(12+4i) ($fp)

    όπου i αύξων αριθμός της παραμέτρου

- par, x, REF, _
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιος βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    ```
    add $t0, $sp, -offset
    sw   $t0, -(12+4i) ($fp)
    ```

  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιος βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    ```
    lw $t0, -offset($sp)
    sw $t0, -(12+4i)($fp)
    ```

  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    ```
    gnlvcode(x)
    sw $t0, -(12+4i) ($fp)
    ```

  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    ```
    gnlvcode(x)
    lw $t0, ($t0)
    sw $t0, -(12+4i) ($fp)
    ```

- par, x, RET, _

  Γεμίζουμε το 3^{ο} πεδίο του εγγραφήματος δραστηριοποίησης της κληθείσας συνάρτησης με τη διεύθυνση της προσωρινής μεταβλητής στην οποία θα επιστραφεί η τιμή

  ```
  add $t0, $sp, -offset
  sw $t0, -8($fp)
  ```

Κλήση Συνάρτησης

- call, _, _, f
  αρχικά γεμίζουμε το 2^{ο} πεδίο του εγγραφήματος δραστηριοποίησης της κληθείσας συνάρτησης με την διεύθυνση του εγγραφήματος δραστηριοποίησης του γονέα της, ώστε η κληθείσα να γνωρίζει που να κοιτάξει αν χρειαστεί να προσπελάσει μία μεταβλητή την οποία έχει δικαίωμα να προσπελάσει, αλλά δεν της ανήκει

  - αν καλούσα και κληθείσα έχουν το ίδιο βάθος φωλιάσματος, τότε έχουν τον ίδιο γονέα

    ```
    lw $t0, -4($sp)
    sw $t0, -4($fp)
    ```

  - αν καλούσα και κληθείσα έχουν διαφορετικό βάθος φωλιάσματος, τότε η καλούσα είναι ο γονέας της κληθείσας

    ```
    sw $sp, -4($fp)
    ```

- Στην συνέχεια μεταφέρουμε τον δείκτη στοίβας στην κληθείσα

  ```
  add $sp, $sp, framelength
  ```

- Καλούμε τη συνάρτηση

  ```
  jal f
  ```

- και όταν επιστρέψουμε παίρνουμε πίσω τον δείκτη στοίβας στην καλούσα

  ```
  add $sp, $sp, -framelength
  ```

- μέσα στην κληθείσα

  - στην αρχή κάθε συνάρτησης αποθηκεύουμε στην πρώτη θέση του εγγραφήματος δραστηριοποίησης την διεύθυνση επιστροφής της την οποία έχει τοποθετήσει στον $ra η jal

    ```
    sw $ra, ($sp)
    ```

  - στο τέλος κάθε συνάρτησης κάνουμε το αντίστροφο, παίρνουμε από την πρώτη θέση του εγγραφήματος δραστηριοποίησης την διεύθυνση επιστροφής της συνάρτησης και την βάζουμε πάλι στον $ra. Μέσω του $ra επιστρέφουμε στην καλούσα

    ```
    lw $ra, ($sp)
    jr $ra
    ```

Αρχή προγράμματος και κυρίως πρόγραμμα

- Το κυρίως πρόγραμμα δεν είναι το πρώτο πράγμα που μεταφράζεται, οπότε στην αρχή του προγράμματος χρειάζεται ένα άλμα που να οδηγεί στην πρώτη ετικέτα του κυρίως προγράμματος

```
j Lmain
```

- Στη συνέχεια πρέπει να κατεβάσουμε τον $sp κατά framelength της main

```
add $sp, $sp –framelength
```

- Και να σημειώσουμε στον $s0 το εγγράφημα δραστηριοποίησης της main ώστε να έχουμε εύκολη πρόσβαση στις global μεταβλητές

```
move $s0, $sp
```

11.TESTS

Για την υλοποίηση της άσκησης , χρειάστηκαν αρκετά τεστ για να σιγουρέψουμε ότι ο κώδικας δούλευε άψογα σε κάθε περίπτωση.

Καθώς η παράδοση έγινε σπαστά σε 3 μέρη, στις προηγούμενες 2 παραδόσεις είχαμε τα ανάλογα τεστ και τα αποτελέσματά τους.

Τώρα ενδεικτικά υπάρχει ένα παράδειγμα για το πως βλέπαμε στον λεκτικό-συντακτικό αναλυτή εάν έπαιρνε σωστά τις λέξεις (word – token) και πήγαινε σε σωστά αναγνωριστικά.

Τα τεστ για λεκτικό- συντακτικό και ενδιάμεσο κώδικα έχουν γίνει turn in, οπότε τώρα ενδεικτικά υπάρχει ένα παράδειγμα για το πως βλέπαμε στον λεκτικό-συντακτικό αναλυτή εάν έπαιρνε σωστά τις λέξεις (word – token) και πήγαινε σε σωστά αναγνωριστικά.

Όλα καλούνται από το τερματικό. Η κατάληξη της γλώσσας μας ορίσαμε πως θα είναι .stl

Παράδειγμα 1 για Λεκτικό

```
program exampleOne
    declare d,i,g,f;

    function two (in g)

        function three (in g, inout x, inandout m)
            declare k, j;
            k:=g;
            j:=g;
            dowhile
                if (k>i) then
                    k:=k-1
                endif;
                j:=j*k;
                k:=k+g
            enddowhile (k<1);
            m:=j;
            return m+1;
            x:=7
        endfunction

        i:=three ( in i+2, inout d, inandout f)
    endfunction

    function one (in g)
        g:=two(in g)
    endfunction

    i:=5;
    g:=1;
    g:=one(in g)

endprogram
```

Άνοιγμα του αρχείου από τερματικό (compiler1.py κώδικας 1ης φάσης)

Όπως βλέπουμε διαβάζει 1-1 χαρακτήρα και ανάλογα την κατάσταση που είναι (βλέπω αυτόματο σελίδα 7).

Έχουν μπει prints σε κάθε βήμα για την επαλήθευση της σωστής αναγνώρισης κάθε λέξης. Σημαντικό πως ενώ διαβάζει τον επόμενο χαρακτήρα της επόμενης λέξης, δεν το χάνει.

```
nova@Nova-Ubuntu:~/Downloads/Compilers-d85dc2305a081f12ca1078b1de35b0a0bdf4f5ee$ python3 compiler1.py firstExampleProfessor.stl
The character is :  p
state =  0
The character is :  r
state =  1
The character is :  o
state =  1
The character is :  g
state =  1
The character is :  r
state =  1
The character is :  a
state =  1
The character is :  m
state =  1
The character is :
state =  1
The word is :  program
The token is 4:  programtk
The character is :  e
state =  0
The character is :  x
state =  1
The character is :  a
state =  1
The character is :  m
state =  1
The character is :  p
state =  1
The character is :  l
state =  1
The character is :  e
state =  1
The character is :  O
state =  1
The character is :  n
state =  1
The character is :  e
state =  1
The character is :
```

Στην συνέχεια παραλείπεται όλο το τερματικό καθώς αφού διαβάζει γράμμα - γράμμα είναι πάρα πολύ μεγάλο. Στην συνέχεια είναι το τέλος του τερματικού , όταν βρίσκει EOF (End Of File), και μας ενημερώνει για το μήκος του κώδικα.

```
state =  0
The word is :  )
The token is 2:  rightPartk
The character is :  e
state =  0
The character is :  n
state =  1
The character is :  d
state =  1
The character is :  p
state =  1
The character is :  r
state =  1
The character is :  o
state =  1
The character is :  g
state =  1
The character is :  r
state =  1
The character is :  a
state =  1
The character is :  m
state =  1
The character is :
state =  1
The word is :  endprogram
The token is 4:  endprogramtk
The character is :
state =  0
---------- End Of File ---------- lines :  33
nova@Nova-Ubuntu:~/Downloads/Compilers-d85dc2305a081f12ca1078b1de35b0a0bdf4f5ee$
```

Τελικός κώδικας

Για την καλύτερη κατανάλωση της λειτουργείας του τελικού κώδικα , όταν τρέχει ο κώδικας δημιουργεί ένα .c αρχείο και ένα .int αρχείο. Αυτά τα 2 είναι προαιρετικά της άσκησης, αλλά μας δίνουν μία πολύ καλή εικόνα του αν δουλεύει καλά ο κώδικας μας.

Φυσικά δημιουργείται και το .asm αρχείο που είναι και ο σκοπός του μεταφραστή μας.

Στην συνέχεια ακολουθούν κάποια παραδείγματα με διαφορετικές λειτουργίες που ελέγχθηκαν για την ορθότητα τους. Διαδέχθηκαν με τρόπο έτσι ώστε να καλύπτουν –αν όχι όλο – μεγάλο μέρος κάθε περίπτωσης – υποπερίπτωσης. Από απλό προς περίπλοκο.

Οι παρακάτω κώδικες μπορούν να βρεθούν στα αρχεία που δημιουργεί όταν τρέξουμε το παράδειγμα

Example 1

Αρχικός Κώδικας

```
1  program examsDokimi1
2
3      declare c,a,b,t;
4
5      function one(inout c)
6          b:=8;
7          c:=a*1;
8          return 0
9          endfunction
10
11     a:=2;
12     t:=one(inout a);
13     b:=c/1;
14     t:=a;
15     a:=2
16
17 endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'one', '_', '_']
1 : [':=', '8', '_', 'b']
2 : ['*', 'a', '1', 'T_0']
3 : [':=', 'T_0', '_', 'c']
4 : ['retv', '0', '_', '_']
5 : ['end_block', 'one', '_', '_']
6 : ['begin_block', 'examsDokimi1', '_', '_']
7 : [':=', '2', '_', 'a']
8 : ['par', 'a', 'REF', '_']
9 : ['par', 'T_1', 'RET', '_']
10 : ['call', 'one', '_', '_']
11 : [':=', 'T_1', '_', 't']
12 : ['/', 'c', '1', 'T_2']
13 : [':=', 'T_2', '_', 'b']
14 : [':=', 'a', '_', 't']
15 : [':=', '2', '_', 'a']
16 : ['halt', '_', '_', '_']
17 : ['end_block', 'examsDokimi1', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example1.stl
['begin_block', 'one', '_', '_']
[':=', '8', '_', 'b']
['*', 'a', '1', 'T_0']
[':=', 'T_0', '_', 'c']
['retv', '0', '_', '_']
['end_block', 'one', '_', '_']
Scope can see :
        The entity has [name , type , offset ]: -->>  ['c', 'inout', 12]
        The entity has [name , type , offset ]: -->>  ['T_0', 16]
['begin_block', 'examsDokimi1', '_', '_']
[':=', '2', '_', 'a']
['par', 'a', 'REF', '_']
['par', 'T_1', 'RET', '_']
['call', 'one', '_', '_']
[':=', 'T_1', '_', 't']
['/', 'c', '1', 'T_2']
[':=', 'T_2', '_', 'b']
[':=', 'a', '_', 't']
[':=', '2', '_', 'a']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['one', ['c', 'inout'], 'func', 20]
        The entity has [name , type , offset ]: -->>  ['T_1', 28]
        The entity has [name , type , offset ]: -->>  ['T_2', 32]
Τελικός Κώδικας
 Framelength :  36
['halt', '_', '_', '_']
['end_block', 'examsDokimi1', '_', '_']
---------- End Of File ---------- lines :  17
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
L_one:
L_0:
    sw $ra, ($sp)
L_1:
    li $t1, 8
    sw $t1, -20($s0)
L_2:
    lw $t1, -16($s0)
    li $t2, 1
    mul $t1, $t1, $t2
L_3:
    lw $t0, -12($sp)
    sw $t1, ($t0)
L_4:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_5:
    lw, $ra, ($sp)
    jr $ra
Lmain:
    addi $sp, $sp, 36
    move $s0, $sp
L_7:
    li $t1, 2
    sw $t1, -16($s0)
L_8:
    addi $fp , $sp , 20
    add $t0 , $sp , -16
    sw $t0 , -12($fp)
L_9:
    add $t0 , $sp , -28
    sw $t0 , -8($fp)
L_10:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 20
    jal L_one
    add $sp, $sp, -20
L_11:
    lw $t1, -28($s0)
    sw $t1, -24($s0)
L_12:
    lw $t1, -12($s0)
    li $t2, 1
    div $t1, $t1, $t2
    sw $t1, -32($s0)
L_13:
    lw $t1, -32($s0)
    sw $t1, -20($s0)
L_14:
    lw $t1, -16($s0)
    sw $t1, -24($s0)
L_15:
    li $t1, 2
    sw $t1, -16($s0)
L_16:

```

Παράδειγμα 2

Αρχικός Κώδικας

```
program examsDokimi2
    declare c,a,b,t;

    t:=2;
    c:=t+1;

    if (not [t=1]) then
        b:=4+5;
        t:=1;
        c:=2
    else
        c:=1;
        a:=5
        endif

endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'examsDokimi2', '_', '_']
1 : [':=', '2', '_', 't']
2 : ['+', 't', '1', 'T_0']
3 : [':=', 'T_0', '_', 'c']
4 : ['=', 't', '1', 11]
5 : ['jump', '_', '_', 6]
6 : ['+', '4', '5', 'T_1']
7 : [':=', 'T_1', '_', 'b']
8 : [':=', '1', '_', 't']
9 : [':=', '2', '_', 'c']
10 : ['jump', '_', '_', 13]
11 : [':=', '1', '_', 'c']
12 : [':=', '5', '_', 'a']
13 : ['halt', '_', '_', '_']
14 : ['end_block', 'examsDokimi2', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example2.stl
['begin_block', 'examsDokimi2', '_', '_']
[':=', '2', '_', 't']
['+', 't', '1', 'T_0']
[':=', 'T_0', '_', 'c']
['=', 't', '1', '_']
['jump', '_', '_', '_']
['+', '4', '5', 'T_1']
[':=', 'T_1', '_', 'b']
[':=', '1', '_', 't']
[':=', '2', '_', 'c']
['jump', '_', '_', '_']
[':=', '1', '_', 'c']
[':=', '5', '_', 'a']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['T_0', 28]
        The entity has [name , type , offset ]: -->>  ['T_1', 32]
 Framelength :  36
['halt', '_', '_', '_']
['end_block', 'examsDokimi2', '_', '_']
---------- End Of File ---------- lines :  16
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 36
    move $s0, $sp
L_1:
    li $t1, 2
    sw $t1, -24($s0)
L_2:
    lw $t1, -24($s0)
    li $t2, 1
    add $t1, $t1, $t2
    sw $t1, -28($s0)
L_3:
    lw $t1, -28($s0)
    sw $t1, -12($s0)
L_4:
    lw $t1, -24($s0)
    li $t2, 1
    beq $t1, $t2, L_11
L_5:
    j L_6
L_6:
    li $t1, 4
    li $t2, 5
    add $t1, $t1, $t2
    sw $t1, -32($s0)
L_7:
    lw $t1, -32($s0)
    sw $t1, -20($s0)
L_8:
    li $t1, 1
    sw $t1, -24($s0)
L_9:
    li $t1, 2
    sw $t1, -12($s0)
L_10:
    j L_13
L_11:
    li $t1, 1
    sw $t1, -12($s0)
L_12:
    li $t1, 5
    sw $t1, -16($s0)
L_13:

```

Παράδειγμα 3

Αρχικός Κώδικας

```
1   program examsDokimi3
2       declare a,b,t;
3
4       b:=5;
5       t:=1;
6       a:=9;
7
8       while (a<=10 or b>2)
9           b:=0;
10          t:=t+1;
11              a:=a+1
12      endwhile
13
14  endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'examsDokimi3', '_', '_']
1 : [':=', '5', '_', 'b']
2 : [':=', '1', '_', 't']
3 : [':=', '9', '_', 'a']
4 : ['<=', 'a', '10', 8]
5 : ['jump', '_', '_', 6]
6 : ['>', 'b', '2', 8]
7 : ['jump', '_', '_', 14]
8 : [':=', '0', '_', 'b']
9 : ['+', 't', '1', 'T_0']
10 : [':=', 'T_0', '_', 't']
11 : ['+', 'a', '1', 'T_1']
12 : [':=', 'T_1', '_', 'a']
13 : ['jump', '_', '_', '4']
14 : ['halt', '_', '_', '_']
15 : ['end_block', 'examsDokimi3', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example3.stl
['begin_block', 'examsDokimi3', '_', '_']
[':=', '5', '_', 'b']
[':=', '1', '_', 't']
[':=', '9', '_', 'a']
['<=', 'a', '10', '_']
['jump', '_', '_', '_']
['>', 'b', '2', '_']
['jump', '_', '_', '_']
[':=', '0', '_', 'b']
['+', 't', '1', 'T_0']
[':=', 'T_0', '_', 't']
['+', 'a', '1', 'T_1']
[':=', 'T_1', '_', 'a']
['jump', '_', '_', '4']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['a', 12]
        The entity has [name , type , offset ]: -->>  ['b', 16]
        The entity has [name , type , offset ]: -->>  ['t', 20]
        The entity has [name , type , offset ]: -->>  ['T_0', 24]
        The entity has [name , type , offset ]: -->>  ['T_1', 28]
 Framelength :  32
['halt', '_', '_', '_']
['end_block', 'examsDokimi3', '_', '_']
---------- End Of File ---------- lines :  14
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 32
    move $s0, $sp
L_1:
    li $t1, 5
    sw $t1, -16($s0)
L_2:
    li $t1, 1
    sw $t1, -20($s0)
L_3:
    li $t1, 9
    sw $t1, -12($s0)
L_4:
    lw $t1, -12($s0)
    li $t2, 10
    ble $t1, $t2, L_8
L_5:
    j L_6
L_6:
    lw $t1, -16($s0)
    li $t2, 2
    bgt $t1, $t2, L_8
L_7:
    j L_14
L_8:
    li $t1, 0
    sw $t1, -16($s0)
L_9:
    lw $t1, -20($s0)
    li $t2, 1
    add $t1, $t1, $t2
    sw $t1, -24($s0)
L_10:
    lw $t1, -24($s0)
    sw $t1, -20($s0)
L_11:
    lw $t1, -12($s0)
    li $t2, 1
    add $t1, $t1, $t2
    sw $t1, -28($s0)
L_12:
    lw $t1, -28($s0)
    sw $t1, -12($s0)
L_13:
    j L_4
L_14:

```

Παράδειγμα 4

Αρχικός Κώδικας

```
program examsDokimi4
    declare c,a,b,t;

   t:=1;
   a:=2;

   dowhile
    b:=a+3;
    t:=t+1
   enddowhile(t<=3)

endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'examsDokimi4', '_', '_']
1 : [':=', '1', '_', 't']
2 : [':=', '2', '_', 'a']
3 : ['+', 'a', '3', 'T_0']
4 : [':=', 'T_0', '_', 'b']
5 : ['+', 't', '1', 'T_1']
6 : [':=', 'T_1', '_', 't']
7 : ['<=', 't', '3', 3]
8 : ['jump', '_', '_', 9]
9 : ['halt', '_', '_', '_']
10 : ['end_block', 'examsDokimi4', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example4.stl
['begin_block', 'examsDokimi4', '_', '_']
[':=', '1', '_', 't']
[':=', '2', '_', 'a']
['+', 'a', '3', 'T_0']
[':=', 'T_0', '_', 'b']
['+', 't', '1', 'T_1']
[':=', 'T_1', '_', 't']
['<=', 't', '3', '_']
['jump', '_', '_', '_']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['T_0', 28]
        The entity has [name , type , offset ]: -->>  ['T_1', 32]
 Framelength :  36
['halt', '_', '_', '_']
['end_block', 'examsDokimi4', '_', '_']
---------- End Of File ---------- lines :  12
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 36
    move $s0, $sp
L_1:
    li $t1, 1
    sw $t1, -24($s0)
L_2:
    li $t1, 2
    sw $t1, -16($s0)
L_3:
    lw $t1, -16($s0)
    li $t2, 3
    add $t1, $t1, $t2
    sw $t1, -28($s0)
L_4:
    lw $t1, -28($s0)
    sw $t1, -20($s0)
L_5:
    lw $t1, -24($s0)
    li $t2, 1
    add $t1, $t1, $t2
    sw $t1, -32($s0)
L_6:
    lw $t1, -32($s0)
    sw $t1, -24($s0)
L_7:
    lw $t1, -24($s0)
    li $t2, 3
    ble $t1, $t2, L_3
L_8:
    j L_9
L_9:
```

Παράδειγμα 5

Αρχικός Κώδικας

```
program examsDokimi5
    declare c,a,b,t;

    b:=10;
    t:=1;
    a:=3;

    incase
    when (a=3):
        t:=4+7;
        c:=2
    when (t>5):
        b:=10-b;
        a:=1;
        t:=1
    endincase

endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'examsDokimi5', '_', '_']
1 : [':=', '10', '_', 'b']
2 : [':=', '1', '_', 't']
3 : [':=', '3', '_', 'a']
4 : [':=', '0', '_', 'T_0']
5 : ['=', 'a', '3', 7]
6 : ['jump', '_', '_', 11]
7 : ['+', '4', '7', 'T_1']
8 : [':=', 'T_1', '_', 't']
9 : [':=', '2', '_', 'c']
10 : [':=', '1', '_', 'T_0']
11 : ['>', 't', '5', 13]
12 : ['jump', '_', '_', 18]
13 : ['-', '10', 'b', 'T_2']
14 : [':=', 'T_2', '_', 'b']
15 : [':=', '1', '_', 'a']
16 : [':=', '1', '_', 't']
17 : [':=', '1', '_', 'T_0']
18 : ['=', 'T_0', '1', '4']
19 : ['halt', '_', '_', '_']
20 : ['end_block', 'examsDokimi5', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example5.stl
['begin_block', 'examsDokimi5', '_', '_']
[':=', '10', '_', 'b']
[':=', '1', '_', 't']
[':=', '3', '_', 'a']
[':=', '0', '_', 'T_0']
['=', 'a', '3', '_']
['jump', '_', '_', '_']
['+', '4', '7', 'T_1']
[':=', 'T_1', '_', 't']
[':=', '2', '_', 'c']
[':=', '1', '_', 'T_0']
['>', 't', '5', '_']
['jump', '_', '_', '_']
['-', '10', 'b', 'T_2']
[':=', 'T_2', '_', 'b']
[':=', '1', '_', 'a']
[':=', '1', '_', 't']
[':=', '1', '_', 'T_0']
['=', 'T_0', '1', '4']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['T_0', 28]
        The entity has [name , type , offset ]: -->>  ['T_1', 32]
        The entity has [name , type , offset ]: -->>  ['T_2', 36]
 Framelength :  40
['halt', '_', '_', '_']
['end_block', 'examsDokimi5', '_', '_']
---------- End Of File ---------- lines :  16
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 40
    move $s0, $sp
L_1:
    li $t1, 10
    sw $t1, -20($s0)
L_2:
    li $t1, 1
    sw $t1, -24($s0)
L_3:
    li $t1, 3
    sw $t1, -16($s0)
L_4:
    li $t1, 0
    sw $t1, -28($s0)
L_5:
    lw $t1, -16($s0)
    li $t2, 3
    beq $t1, $t2, L_7
L_6:
    j L_11
L_7:
    li $t1, 4
    li $t2, 7
    add $t1, $t1, $t2
    sw $t1, -32($s0)
L_8:
    lw $t1, -32($s0)
    sw $t1, -24($s0)
L_9:
    li $t1, 2
    sw $t1, -12($s0)
L_10:
    li $t1, 1
    sw $t1, -28($s0)
```

```
L_11:
    lw $t1, -24($s0)
    li $t2, 5
    bgt $t1, $t2, L_13
L_12:
    j L_18
L_13:
    li $t1, 10
    lw $t2, -20($s0)
    sub $t1, $t1, $t2
    sw $t1, -36($s0)
L_14:
    lw $t1, -36($s0)
    sw $t1, -20($s0)
L_15:
    li $t1, 1
    sw $t1, -16($s0)
L_16:
    li $t1, 1
    sw $t1, -24($s0)
L_17:
    li $t1, 1
    sw $t1, -28($s0)
L_18:
    lw $t1, -28($s0)
    li $t2, 1
    beq $t1, $t2, L_4
L_19:
```

Παράδειγμα 6

Αρχικός Κώδικας

```
program examsDokimi6
    declare c,a,b,t;

    b:=1;
    t:=10;

    loop
    b:=5-1;
    a:=1;
    t:=t-1;
    exit;
    b:=t+a
    endloop

endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'examsDokimi6', '_', '_']
1 : [':=', '1', '_', 'b']
2 : [':=', '10', '_', 't']
3 : ['-', '5', '1', 'T_0']
4 : [':=', 'T_0', '_', 'b']
5 : [':=', '1', '_', 'a']
6 : ['-', 't', '1', 'T_1']
7 : [':=', 'T_1', '_', 't']
8 : ['jump', '_', '_', 12]
9 : ['+', 't', 'a', 'T_2']
10 : [':=', 'T_2', '_', 'b']
11 : ['jump', '_', '_', '3']
12 : ['halt', '_', '_', '_']
13 : ['end_block', 'examsDokimi6', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example6.stl
['begin_block', 'examsDokimi6', '_', '_']
[':=', '1', '_', 'b']
[':=', '10', '_', 't']
['-', '5', '1', 'T_0']
[':=', 'T_0', '_', 'b']
[':=', '1', '_', 'a']
['-', 't', '1', 'T_1']
[':=', 'T_1', '_', 't']
['jump', '_', '_', '_']
['+', 't', 'a', 'T_2']
[':=', 'T_2', '_', 'b']
['jump', '_', '_', '3']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['T_0', 28]
        The entity has [name , type , offset ]: -->>  ['T_1', 32]
        The entity has [name , type , offset ]: -->>  ['T_2', 36]
 Framelength :  40
['halt', '_', '_', '_']
['end_block', 'examsDokimi6', '_', '_']
---------- End Of File ---------- lines :  15
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 40
    move $s0, $sp
L_1:
    li $t1, 1
    sw $t1, -20($s0)
L_2:
    li $t1, 10
    sw $t1, -24($s0)
L_3:
    li $t1, 5
    li $t2, 1
    sub $t1, $t1, $t2
    sw $t1, -28($s0)
L_4:
    lw $t1, -28($s0)
    sw $t1, -20($s0)
L_5:
    li $t1, 1
    sw $t1, -16($s0)
L_6:
    lw $t1, -24($s0)
    li $t2, 1
    sub $t1, $t1, $t2
    sw $t1, -32($s0)
L_7:
    lw $t1, -32($s0)
    sw $t1, -24($s0)
L_8:
    j L_12
L_9:
    lw $t1, -24($s0)
    lw $t2, -16($s0)
    add $t1, $t1, $t2
    sw $t1, -36($s0)
L_10:
    lw $t1, -36($s0)
    sw $t1, -20($s0)
L_11:
    j L_3
L_12:
```

Παράδειγμα 7

Αρχικός Κώδικας

```
program example1
    declare d,i,g,f;

    function two (in g)

        function three (in g, inout x, inandout m)

            declare k, j;
            k:=g;
            j:=g;

            dowhile
                if (k>i) then
                    k:=k-1
                endif

            enddowhile (k<1);

        m:=j;
            return m+1;
            x:=7
        endfunction

        i:=three (in i+2, inout d, inandout f);
        return 0
    endfunction

    function one (in g)
        g:=two(in g);
        return 0
    endfunction

    i:=5;
    g:=1;
    g:=one(in g)
endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'three', '_', '_']
1 : [':=', 'g', '_', 'k']
2 : [':=', 'g', '_', 'j']
3 : ['>', 'k', 'i', 5]
4 : ['jump', '_', '_', 8]
5 : ['-', 'k', '1', 'T_0']
6 : [':=', 'T_0', '_', 'k']
7 : ['jump', '_', '_', 8]
8 : ['<', 'k', '1', 3]
9 : ['jump', '_', '_', 10]
10 : [':=', 'j', '_', 'm']
11 : ['+', 'm', '1', 'T_1']
12 : ['retv', 'T_1', '_', '_']
13 : [':=', '7', '_', 'x']
14 : ['end_block', 'three', '_', '_']
15 : ['begin_block', 'two', '_', '_']
16 : ['+', 'i', '2', 'T_2']
17 : ['par', 'T_2', 'CV', '_']
18 : ['par', 'd', 'REF', '_']
19 : ['par', 'f', 'CP', '_']
20 : ['par', 'T_3', 'RET', '_']
21 : ['call', 'three', '_', '_']
22 : [':=', 'T_3', '_', 'i']
23 : ['retv', '0', '_', '_']
24 : ['end_block', 'two', '_', '_']
25 : ['begin_block', 'one', '_', '_']
26 : ['par', 'g', 'CV', '_']
27 : ['par', 'T_4', 'RET', '_']
28 : ['call', 'two', '_', '_']
29 : [':=', 'T_4', '_', 'g']
30 : ['retv', '0', '_', '_']
31 : ['end_block', 'one', '_', '_']
32 : ['begin_block', 'example1', '_', '_']
33 : [':=', '5', '_', 'i']
34 : [':=', '1', '_', 'g']
35 : ['par', 'g', 'CV', '_']
36 : ['par', 'T_5', 'RET', '_']
37 : ['call', 'one', '_', '_']
38 : [':=', 'T_5', '_', 'g']
39 : ['halt', '_', '_', '_']
40 : ['end_block', 'example1', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example6.stl
['begin_block', 'examsDokimi6', '_', '_']
[':=', '1', '_', 'b']
[':=', '10', '_', 't']
['-', '5', '1', 'T_0']
[':=', 'T_0', '_', 'b']
[':=', '1', '_', 'a']
['-', 't', '1', 'T_1']
[':=', 'T_1', '_', 't']
['jump', '_', '_', '_']
['+', 't', 'a', 'T_2']
[':=', 'T_2', '_', 'b']
['jump', '_', '_', '3']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['c', 12]
        The entity has [name , type , offset ]: -->>  ['a', 16]
        The entity has [name , type , offset ]: -->>  ['b', 20]
        The entity has [name , type , offset ]: -->>  ['t', 24]
        The entity has [name , type , offset ]: -->>  ['T_0', 28]
        The entity has [name , type , offset ]: -->>  ['T_1', 32]
        The entity has [name , type , offset ]: -->>  ['T_2', 36]
 Framelength :  40
['halt', '_', '_', '_']
['end_block', 'examsDokimi6', '_', '_']
---------- End Of File ---------- lines :  15
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
L_three:
L_0:
    sw $ra, ($sp)
L_1:
    lw $t1, -12($sp)
L_2:
    lw $t1, -12($sp)
L_3:
    lw $t2, -16($s0)
    bgt $t1, $t2, L_5
L_4:
    j L_8
L_5:
    li $t2, 1
    sub $t1, $t1, $t2
L_6:
L_7:
    j L_8
L_8:
    li $t2, 1
    blt $t1, $t2, L_3
L_9:
    j L_10
L_10:
    sw $t1, -24($sp)
    sw $t1, -24($sp)
    sw $t1, ($t0)
L_11:
    lw $t1, -24($sp)
    lw $t1, ($t0)
    li $t2, 1
    add $t1, $t1, $t2
L_12:
    lw $t0, -8($sp)
    sw $t1, ($t0)
```

```
L_13:
    li $t1, 7
    lw $t0, -20($sp)
    sw $t1, ($t0)
L_14:
    lw, $ra, ($sp)
    jr $ra
L_two:
L_15:
    sw $ra, ($sp)
L_16:
    lw $t1, -16($s0)
    li $t2, 2
    add $t1, $t1, $t2
L_17:
    addi $fp , $sp , 44
    sw $t0, -12($fp)
L_18:
    lw $t0 , -4($sp)
    add $t0 , $t0 , -12
    sw $t0 , -16($fp)
L_19:
    lw $t0 , -4($sp)
    add $t0 , $t0 , -24
    lw $t0 , ($t0)
    sw $t0 , -20($fp)
L_20:
    add $t0 , $sp , -20
    sw $t0 , -8($fp)
L_21:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 44
    jal L_three
    add $sp, $sp, -44
L_22:
    sw $t1, -16($s0)
L_23:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_24:
    lw, $ra, ($sp)
    jr $ra
L_one:
```

```
L_25:
    sw $ra, ($sp)
L_26:
    addi $fp , $sp , 24
    lw $t0, -12($sp)
    sw $t0, -12($fp)
L_27:
    add $t0 , $sp , -32
    sw $t0 , -8($fp)
L_28:
    sw $sp , -4($fp)
    add $sp, $sp, 24
    jal L_two
    add $sp, $sp, -24
L_29:
    sw $t1, -12($sp)
L_30:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_31:
    lw, $ra, ($sp)
    jr $ra
Lmain:
    addi $sp, $sp, 32
    move $s0, $sp
L_33:
    li $t1, 5
    sw $t1, -16($s0)
L_34:
    li $t1, 1
    sw $t1, -20($s0)
L_35:
    addi $fp , $sp , 36
    lw $t0, -20($s0)
    sw $t0, -12($fp)
L_36:
    add $t0 , $sp , -28
    sw $t0 , -8($fp)
L_37:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 36
    jal L_one
    add $sp, $sp, -36
L_38:
    lw $t1, -28($s0)
    sw $t1, -20($s0)
L_39:
```

Παράδειγμα 8

Αρχικός Κώδικας

```
program example2
declare x,y,z;
function p1(in x, inout z, inout v)
declare w;
function p2(inout z)
declare q;
function p3(inout a, inout b)
declare k;
if (v<>0) then
v:=z+b;
a:=1
else
a:=v/b
endif;
k:=x;
return 0
endfunction
q:=y+w;
z:=q*x;
v:= p3(inout q, inout v);
return 0
endfunction
if (x<y) then
w:=x+y
else
w:=x*y
endif;
z:= p2(inout z);
return 0
endfunction
x:=1;
y:=2;
z:= p1(in x+y, inout z, inout y)
endprogram
```

Ενδιάμεσος Κώδικας

```
0 : ['begin_block', 'p3', '_', '_']
1 : ['<>', 'v', '0', 3]
2 : ['jump', '_', '_', 7]
3 : ['+', 'z', 'b', 'T_0']
4 : [':=', 'T_0', '_', 'v']
5 : [':=', '1', '_', 'a']
6 : ['jump', '_', '_', 9]
7 : ['/', 'v', 'b', 'T_1']
8 : [':=', 'T_1', '_', 'a']
9 : [':=', 'x', '_', 'k']
10 : ['retv', '0', '_', '_']
11 : ['end_block', 'p3', '_', '_']
12 : ['begin_block', 'p2', '_', '_']
13 : ['+', 'y', 'w', 'T_2']
14 : [':=', 'T_2', '_', 'q']
15 : ['*', 'q', 'x', 'T_3']
16 : [':=', 'T_3', '_', 'z']
17 : ['par', 'q', 'REF', '_']
18 : ['par', 'v', 'REF', '_']
19 : ['par', 'T_4', 'RET', '_']
20 : ['call', 'p3', '_', '_']
21 : [':=', 'T_4', '_', 'v']
22 : ['retv', '0', '_', '_']
23 : ['end_block', 'p2', '_', '_']
24 : ['begin_block', 'p1', '_', '_']
25 : ['<', 'x', 'y', 27]
26 : ['jump', '_', '_', 30]
27 : ['+', 'x', 'y', 'T_5']
28 : [':=', 'T_5', '_', 'w']
29 : ['jump', '_', '_', 32]
30 : ['*', 'x', 'y', 'T_6']
31 : [':=', 'T_6', '_', 'w']
32 : ['par', 'z', 'REF', '_']
33 : ['par', 'T_7', 'RET', '_']
34 : ['call', 'p2', '_', '_']
35 : [':=', 'T_7', '_', 'z']
36 : ['retv', '0', '_', '_']
37 : ['end_block', 'p1', '_', '_']
38 : ['begin_block', 'example2', '_', '_']
39 : [':=', '1', '_', 'x']
40 : [':=', '2', '_', 'y']
41 : ['+', 'x', 'y', 'T_8']
42 : ['par', 'T_8', 'CV', '_']
43 : ['par', 'z', 'REF', '_']
44 : ['par', 'y', 'REF', '_']
45 : ['par', 'T_9', 'RET', '_']
46 : ['call', 'p1', '_', '_']
47 : [':=', 'T_9', '_', 'z']
48 : ['halt', '_', '_', '_']
49 : ['end_block', 'example2', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example8.stl
['begin_block', 'p3', '_', '_']
['<>', 'v', '0', '_']
['jump', '_', '_', '_']
['+', 'z', 'b', 'T_0']
[':=', 'T_0', '_', 'v']
[':=', '1', '_', 'a']
['jump', '_', '_', '_']
['/', 'v', 'b', 'T_1']
[':=', 'T_1', '_', 'a']
[':=', 'x', '_', 'k']
['retv', '0', '_', '_']
['end_block', 'p3', '_', '_']
Scope can see :
        The entity has [name , type , offset ]: -->>  ['x', 'in', 12]
        The entity has [name , type , offset ]: -->>  ['z', 'inout', 16]
        The entity has [name , type , offset ]: -->>  ['v', 'inout', 20]
        The entity has [name , type , offset ]: -->>  ['z', 'inout', 24]
        The entity has [name , type , offset ]: -->>  ['a', 'inout', 28]
        The entity has [name , type , offset ]: -->>  ['b', 'inout', 32]
        The entity has [name , type , offset ]: -->>  ['k', 36]
        The entity has [name , type , offset ]: -->>  ['T_0', 40]
        The entity has [name , type , offset ]: -->>  ['T_1', 44]
['begin_block', 'p2', '_', '_']
['+', 'y', 'w', 'T_2']
[':=', 'T_2', '_', 'q']
['*', 'q', 'x', 'T_3']
[':=', 'T_3', '_', 'z']
['par', 'q', 'REF', '_']
['par', 'v', 'REF', '_']
['par', 'T_4', 'RET', '_']
['call', 'p3', '_', '_']
[':=', 'T_4', '_', 'v']
['retv', '0', '_', '_']
['end_block', 'p2', '_', '_']
Scope can see :
        The entity has [name , type , offset ]: -->>  ['x', 'in', 12]
        The entity has [name , type , offset ]: -->>  ['z', 'inout', 16]
        The entity has [name , type , offset ]: -->>  ['v', 'inout', 20]
        The entity has [name , type , offset ]: -->>  ['z', 'inout', 24]
        The entity has [name , type , offset ]: -->>  ['q', 28]
        The entity has [name , type , offset ]: -->>  ['p3', ['x', 'in'], ['z', 'inout'], ['v', 'inout'], ['z', 'inout'], ['a', 'inout'], ['b', 'inout'], 'func', 48]
        The entity has [name , type , offset ]: -->>  ['T_2', 32]
        The entity has [name , type , offset ]: -->>  ['T_3', 36]
        The entity has [name , type , offset ]: -->>  ['T_4', 40]
['begin_block', 'p1', '_', '_']
['<', 'x', 'y', '_']
['jump', '_', '_', '_']
['+', 'x', 'y', 'T_5']
[':=', 'T_5', '_', 'w']
['jump', '_', '_', '_']
['*', 'x', 'y', 'T_6']
[':=', 'T_6', '_', 'w']
['par', 'z', 'REF', '_']
['par', 'T_7', 'RET', '_']
['call', 'p2', '_', '_']
[':=', 'T_7', '_', 'z']
['retv', '0', '_', '_']
['end_block', 'p1', '_', '_']
Scope can see :
        The entity has [name , type , offset ]: -->>  ['x', 'in', 12]
        The entity has [name , type , offset ]: -->>  ['z', 'inout', 16]
        The entity has [name , type , offset ]: -->>  ['v', 'inout', 20]
        The entity has [name , type , offset ]: -->>  ['w', 24]
        The entity has [name , type , offset ]: -->>  ['p2', ['x', 'in'], ['z', 'inout'], ['v', 'inout'], ['z', 'inout'], 'func', 44]
        The entity has [name , type , offset ]: -->>  ['T_5', 28]
        The entity has [name , type , offset ]: -->>  ['T_6', 32]
        The entity has [name , type , offset ]: -->>  ['T_7', 36]
['begin_block', 'example2', '_', '_']
[':=', '1', '_', 'x']
[':=', '2', '_', 'y']
['+', 'x', 'y', 'T_8']
['par', 'T_8', 'CV', '_']
['par', 'z', 'REF', '_']
['par', 'y', 'REF', '_']
['par', 'T_9', 'RET', '_']
['call', 'p1', '_', '_']
[':=', 'T_9', '_', 'z']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['x', 12]
        The entity has [name , type , offset ]: -->>  ['y', 16]
        The entity has [name , type , offset ]: -->>  ['z', 20]
        The entity has [name , type , offset ]: -->>  ['p1', ['x', 'in'], ['z', 'inout'], ['v', 'inout'], 'func', 40]
        The entity has [name , type , offset ]: -->>  ['T_8', 24]
        The entity has [name , type , offset ]: -->>  ['T_9', 28]
 Framelength :  32
['halt', '_', '_', '_']
['end_block', 'example2', '_', '_']
---------- End Of File ---------- lines :  34
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
L_p3:
L_0:
    sw $ra, ($sp)
L_1:
    lw $t0, -20($sp)
    lw $t1, ($t0)
    li $t2, 0
    bne $t1, $t2, L_3
L_2:
    j L_7
L_3:
    lw $t0, -16($sp)
    lw $t1, ($t0)
    lw $t0, -32($sp)
    lw $t2, ($t0)
    add $t1, $t1, $t2
L_4:
    lw $t0, -20($sp)
    sw $t1, ($t0)
L_5:
    li $t1, 1
    lw $t0, -28($sp)
    sw $t1, ($t0)
L_6:
    j L_9
L_7:
    lw $t0, -20($sp)
    lw $t1, ($t0)
    lw $t0, -32($sp)
    lw $t2, ($t0)
    div $t1, $t1, $t2
L_8:
    lw $t0, -28($sp)
    sw $t1, ($t0)
L_9:
    lw $t1, -12($sp)
L_10:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_11:
    lw, $ra, ($sp)
    jr $ra
L_p2:
L_12:
    sw $ra, ($sp)
L_13:
    lw $t1, -16($s0)
    add $t1, $t1, $t2
L_14:
L_15:
    lw $t2, -12($sp)
    mul $t1, $t1, $t2
L_16:
    lw $t0, -16($sp)
    sw $t1, ($t0)
L_17:
    addi $fp , $sp , 48
    add $t0 , $sp , -28
    sw $t0 , -12($fp)
L_18:
    lw $t0 , -20($sp)
    sw $t0 , -16($fp)
L_19:
    add $t0 , $sp , -40
    sw $t0 , -8($fp)
L_20:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 48
    jal L_p3
    add $sp, $sp, -48
L_21:
    lw $t0, -20($sp)
    sw $t1, ($t0)
L_22:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_23:
    lw, $ra, ($sp)
    jr $ra
L_p1:
L_24:
    sw $ra, ($sp)
L_25:
    lw $t1, -12($sp)
    lw $t2, -16($s0)
    blt $t1, $t2, L_27
L_26:
    j L_30
L_27:
    lw $t1, -12($sp)
    lw $t2, -16($s0)
    add $t1, $t1, $t2
L_28:
L_29:
    j L_32
L_30:
    lw $t1, -12($sp)
    lw $t2, -16($s0)
    mul $t1, $t1, $t2
L_31:
L_32:
    addi $fp , $sp , 44
    lw $t0 , -16($sp)
    sw $t0 , -12($fp)
L_33:
    add $t0 , $sp , -36
    sw $t0 , -8($fp)
L_34:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 44
    jal L_p2
    add $sp, $sp, -44
L_35:
    lw $t0, -16($sp)
    sw $t1, ($t0)
L_36:
    li $t1, 0
    lw $t0, -8($sp)
    sw $t1, ($t0)
L_37:
    lw, $ra, ($sp)
    jr $ra
Lmain:
    addi $sp, $sp, 32
    move $s0, $sp
L_39:
    li $t1, 1
    sw $t1, -12($s0)
L_40:
    li $t1, 2
    sw $t1, -16($s0)
L_41:
    lw $t1, -12($s0)
    lw $t2, -16($s0)
    add $t1, $t1, $t2
    sw $t1, -24($s0)
L_42:
    addi $fp , $sp , 40
    lw $t0, -24($s0)
    sw $t0, -12($fp)
L_43:
    add $t0 , $sp , -20
    sw $t0 , -16($fp)
L_44:
    add $t0 , $sp , -16
    sw $t0 , -20($fp)
L_45:
    add $t0 , $sp , -28
    sw $t0 , -8($fp)
L_46:
    lw $t0 , -4($sp)
    sw $t0 , -4($fp)
    add $sp, $sp, 40
    jal L_p1
    add $sp, $sp, -40
L_47:
```

Παράδειγμα 9

Αρχικός Κώδικας

```
program example3
declare a,b,c,d,e,x,y,px,py,temp;
loop
if (not [a<c and b<d]) then
exit
endif;
if (a=e) then
c:=c+e
else
loop
if (not [a<=d]) then
exit
endif;
a:=a+b
endloop
endif
endloop;
temp:=px;
x:=1;
y:=2
endprogram
```

Ενδιάμεσος Κ

```
0 : ['begin_block', 'example3', '_', '_']
1 : ['<', 'a', 'c', 3]
2 : ['jump', '_', '_', '_']
3 : ['<', 'b', 'd', 7]
4 : ['jump', '_', '_', 5]
5 : ['jump', '_', '_', '_']
6 : ['jump', '_', '_', 7]
7 : ['=', 'a', 'e', 9]
8 : ['jump', '_', '_', 12]
9 : ['+', 'c', 'e', 'T_0']
10 : [':=', 'T_0', '_', 'c']
11 : ['jump', '_', '_', 19]
12 : ['<=', 'a', 'd', 16]
13 : ['jump', '_', '_', 14]
14 : ['jump', '_', '_', 20]
15 : ['jump', '_', '_', 16]
16 : ['+', 'a', 'b', 'T_1']
17 : [':=', 'T_1', '_', 'a']
18 : ['jump', '_', '_', '12']
19 : ['jump', '_', '_', '1']
20 : [':=', 'px', '_', 'temp']
21 : [':=', '1', '_', 'x']
22 : [':=', '2', '_', 'y']
23 : ['halt', '_', '_', '_']
24 : ['end_block', 'example3', '_', '_']
```

Κλήση Τερματικού

```
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$ python3 met.py example9.stl
['begin_block', 'example3', '_', '_']
['<', 'a', 'c', '_']
['jump', '_', '_', '_']
['<', 'b', 'd', '_']
['jump', '_', '_', '_']
['jump', '_', '_', '_']
['jump', '_', '_', '_']
['=', 'a', 'e', '_']
['jump', '_', '_', '_']
['+', 'c', 'e', 'T_0']
[':=', 'T_0', '_', 'c']
['jump', '_', '_', '_']
['<=', 'a', 'd', '_']
['jump', '_', '_', '_']
['jump', '_', '_', '_']
['jump', '_', '_', '_']
['+', 'a', 'b', 'T_1']
[':=', 'T_1', '_', 'a']
['jump', '_', '_', '12']
['jump', '_', '_', '1']
[':=', 'px', '_', 'temp']
[':=', '1', '_', 'x']
[':=', '2', '_', 'y']
Scope can see:
        The entity has [name , type , offset ]: -->>  ['a', 12]
        The entity has [name , type , offset ]: -->>  ['b', 16]
        The entity has [name , type , offset ]: -->>  ['c', 20]
        The entity has [name , type , offset ]: -->>  ['d', 24]
        The entity has [name , type , offset ]: -->>  ['e', 28]
        The entity has [name , type , offset ]: -->>  ['x', 32]
        The entity has [name , type , offset ]: -->>  ['y', 36]
        The entity has [name , type , offset ]: -->>  ['px', 40]
        The entity has [name , type , offset ]: -->>  ['py', 44]
        The entity has [name , type , offset ]: -->>  ['temp', 48]
        The entity has [name , type , offset ]: -->>  ['T_0', 52]
        The entity has [name , type , offset ]: -->>  ['T_1', 56]
 Framelength :  60
['halt', '_', '_', '_']
['end_block', 'example3', '_', '_']
---------- End Of File ---------- lines :  21
nova@Nova-Ubuntu:~/Downloads/Compilers-master/TurnIn$
```

Τελικός Κώδικας

```
#MARS File created by Starlet

.data
.asciiz
.text

    j Lmain
Lmain:
    addi $sp, $sp, 60
    move $s0, $sp
L_1:
    lw $t1, -12($s0)
    lw $t2, -20($s0)
    blt $t1, $t2, L_3
L_2:
    j L__
L_3:
    lw $t1, -16($s0)
    lw $t2, -24($s0)
    blt $t1, $t2, L_7
L_4:
    j L_5
L_5:
    j L__
L_6:
    j L_7
L_7:
    lw $t1, -12($s0)
    lw $t2, -28($s0)
    beq $t1, $t2, L_9
L_8:
    j L_12
L_9:
    lw $t1, -20($s0)
    lw $t2, -28($s0)
    add $t1, $t1, $t2
    sw $t1, -52($s0)
L_10:
    lw $t1, -52($s0)
    sw $t1, -20($s0)
L_11:
    j L_19
```

```
L_12:
    lw $t1, -12($s0)
    lw $t2, -24($s0)
    ble $t1, $t2, L_16
L_13:
    j L_14
L_14:
    j L_20
L_15:
    j L_16
L_16:
    lw $t1, -12($s0)
    lw $t2, -16($s0)
    add $t1, $t1, $t2
    sw $t1, -56($s0)
L_17:
    lw $t1, -56($s0)
    sw $t1, -12($s0)
L_18:
    j L_12
L_19:
    j L_1
L_20:
    lw $t1, -40($s0)
    sw $t1, -48($s0)
L_21:
    li $t1, 1
    sw $t1, -32($s0)
L_22:
    li $t1, 2
    sw $t1, -36($s0)
L_23:
```

12. ΣΧΟΛΙΑ

Η άσκηση έγινε σε python3 , και το assembly αρχείο το τρέχαμε μέσω MIPS.

Το πιο χρονοβόρο κομμάτι ήταν του τελικού κώδικας καθώς και το πιο δύσκολο.

Τρέχουν όλα πάρα πολύ καλά , και αν ξέφυγε κάτι ήταν θέμα χρόνου, καθώς μέχρι και τελευταία στιγμή δεν σταματήσαμε να δουλεύουμε το project.

Αναρτούμε το τελικό python αρχείο (met.py) που περιέχει όλες τις συναρτήσεις και κλήσεις που υλοποιήθηκαν κατά την διάρκεια του εξαμήνου, καθώς και τα test για τον τελικό κώδικα (9 τεστ).