ΜΕΤΑΦΡΑΣΤΕΣ

ΤΕΛΙΚΗ ΑΝΑΦΟΡΑ

ΜΕΛΗ ΟΜΑΔΑΣ:

ΑΛΕΞΑΝΔΡΟΠΟΥΛΟΣ ΔΗΜΗΤΡΙΟΣ Α.Μ. 2928
ΜΠΟΖ ΝΤΟΥΡΑΝ Α.Μ. 2310

Περιγραφή της Γλώσσας Προγραμματισμού Starlet

Η Starlet είναι μια μικρή γλώσσα προγραμματισμού φτιαγμένη με βάση τις ανάγκες της προγραμματιστικής άσκησης του μαθήματος. 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 Starlet, υποστηρίζει συναρτήσεις, μετάδοση παραμέτρων με αναφορά, τιμή και αντιγραφή, αναδρομικές κλήσεις και άλλες ενδιαφέρουσες δομές. Επίσης επιτρέπει φώλιασμα στη δήλωση συναρτήσεων, κάτι που λίγες γλώσσες υποστηρίζουν (όπως η Pascal, αλλά όχι η C).

Από την άλλη πλευρά, η Starlet δεν υποστηρίζει βασικά προγραμματιστικά εργαλεία, όπως η δομή for, ή τύπους δεδομένων όπως οι πραγματικοί αριθμοί και συμβολοσειρές. Οι παραλήψεις αυτές, έχουν γίνει ώστε να απλουστευτεί η διαδικασία κατασκευής του μεταγλωττιστή, μία απλούστευση όμως που έχει να κάνει μόνο με τη μείωση των γραμμών κώδικα και όχι με τη δυσκολία κατασκευής του ή την εκπαιδευτική αξία της άσκησης.

Παρακάτω παρουσιάζεται μια περιγραφή της γλώσσας:

Λεκτικές Μονάδες

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 (<< **/*** >>, << ***/** >>, << **//** >>).

Τα σύμβολα << **[** >> και << **]** >> χρησιμοποιούνται στις λογικές παραστάσεις όπως τα σύμβολα << **(** >> και << **)** >> στις αριθμητικές παραστάσεις.

Μερικές λέξεις είναι δεσμευμένες:

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 μπορεί 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 όλα 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.

Μορφή Προγράμματος

program id
    declarations
    subprograms
    statements
endprogram

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 **declarations.** 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 **declarations.**

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** >>

Δομές της γλώσσας

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

enddowhile, όσο η συνθήκη condition ισχύει. Οι statements εκτελούνται τουλάχιστον μία φορά, πριν αποτιμηθεί η condition.

Επανάληψη loop

loop
    statements
endloop

Η εντολή επανάληψης **loop** επαναλαμβάνει για πάντα τις εντολές statements που βρίσκονται ανάμεσα στο **loop** και στο **endloop.** Έξοδος από το 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 συνέχεια 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 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**, εάν έστω και μία από τις statements έχει εκτελεστεί.

Επιστροφή Τιμής

return expression

Χρησιμοποιείται μέσα σε συναρτήσεις για να επιστραφεί το αποτέλεσμα της συνάρτησης.

Έξοδος

print expression

Εμφανίζει στην οθόνη το αποτέλεσμα της αποτίμησης του expression

Είσοδος

input id

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

Υποπρογράμματα

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 τρεις 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 ολοκληρώνεται η εκτέλεση της συνάρτησης.

Στην κλήση μίας 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 .

Γραμματική της Starlet

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

default : <statements> **enddefault**

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>

Γενική Υλοποίηση Μεταγλωττιστή

Λεκτική Ανάλυση

Η πρώτη φάση ενός μεταγλωττιστή ονομάζεται λεκτική ανάλυση. Ο λεκτικός αναλυτής διαβάζει τη ροή των χαρακτήρων που αποτελούν το πηγαίο πρόγραμμα και ομαδοποιεί τους χαρακτήρες σε ακολουθίες με κάποιο νόημα, που αποκαλούνται λεξήματα. Για κάθε λέξημα, ο λεκτικός αναλυτής παράγει ως έξοδο ένα λεκτικό σύμβολο (token), το οποίο περνά στην επόμενη φάση, τη *συντακτική ανάλυση*. Στο λεκτικό σύμβολο, το πρώτο συστατικό λέγεται *όνομα-λεκτικού συμβόλου* και είναι ένα αφηρημένο σύμβολο που χρησιμοποιείται κατά τη συντακτική ανάλυση, ενώ το δεύτερο συστατικό λέγεται *τιμή-ιδιότητας* και δεικτοδοτεί μία καταχώρηση στον *πίνακα συμβόλων*. Όταν ο λεκτικός αναλυτής ανακαλύπτει ένα λέξημα που αποτελεί έναν προσδιοριστή, χρειάζεται να εισάγει αυτό το λέξημα στον *πίνακα συμβόλων*. Όταν περισσότερα από ένα λεξήματα μπορούν να ταυτιστούν με ένα πρότυπο, ο λεκτικός αναλυτής πρέπει να παρέχει στις μεταγενέστερες φάσεις του μεταγλωττιστή επιπλέον πληροφορία σχετικά με το συγκεκριμένο λέξημα που ταυτίστηκε.

Συντακτική Ανάλυση

Η δεύτερη φάση ενός μεταγλωττιστή είναι η συντακτική ανάλυση. Ο συντακτικός αναλυτής χρησιμοποιεί τα πρώτα συστατικά στοιχεία των λεκτικών συμβόλων που παρήγαγε ο λεκτικός αναλυτής για να δημιουργήσει μια δενδρικού τύπου ενδιάμεση αναπαράσταση που απεικονίζει την γραμματική δομή της ροής των λεκτικών συμβόλων. Μια συνηθισμένη αναπαράσταση είναι το συντακτικό

δέντρο στο οποίο κάθε εσωτερικός κόμβος αναπαριστά μια πράξη και τα παιδιά αυτού του κόμβου αναπαριστούν τα ορίσματα της πράξης. Στο μοντέλο του μεταγλωττιστή μας, ο συντακτικός αναλυτής αποκτά μια ακολουθία από λεκτικές μονάδες από το λεκτικό αναλυτή και επαληθεύει ότι η ακολουθία των λεκτικών μονάδων μπορεί να παραχθεί από τη γραμματική για την πηγαία γλώσσα. Αναμένουμε από το συντακτικό αναλυτή να αναφέρει οποιαδήποτε συντακτικά σφάλματα ώστε να συνεχίσει την επεξεργασία του υπόλοιπου προγράμματος.

Σημασιολογική Ανάλυση

Ο σημασιολογικός αναλυτής χρησιμοποιεί το συντακτικό δέντρο και τις πληροφορίες του *πίνακα συμβόλων* για να ελέγξει αν το πηγαίο πρόγραμμα είναι σημασιολογικά συνεπές με τον ορισμό της γλώσσας. Επίσης συλλέγει πληροφορίες τύπων και τις αποθηκεύει είτε στο συντακτικό δέντρο είτε στον πίνακα συμβόλων, για μελλοντική χρήση κατά τη διάρκεια δημιουργίας του *ενδιάμεσου κώδικα*. Ένα σημαντικό μέρος της σημασιολογικής ανάλυσης είναι ο έλεγχος τύπων όπου ο μεταγλωττιστής ελέγχει ότι κάθε τελεστής έχει ταιριαστούς τελεστέους.

Παραγωγή Ενδιάμεσου Κώδικα

Κατά τη διαδικασία μετάφρασης ενός πηγαίου προγράμματος σε τελικό κώδικα ένας μεταγλωττιστής μπορεί να κατασκευάσει μια ή περισσότερες ενδιάμεσες αναπαραστάσεις, οι οποίες μπορεί να έχουν μια ποικιλία μορφών. Τα συντακτικά δέντρα είναι μια μορφή ενδιάμεσης αναπαράστασης τα οποία χρησιμοποιούνται συνήθως κατά τη διάρκεια της συντακτικής και σημασιολογικής ανάλυσης. Μετά τη συντακτική και σημασιολογική ανάλυση, παράγεται μια συγκεκριμένη ενδιάμεση αναπαράσταση χαμηλού επιπέδου που μπορεί να θεωρηθεί ότι προσομοιάζει έναν τύπο γλώσσας μηχανής,την οποία μπορούμε να φανταστούμε ως ένα πρόγραμμα για μια αφηρημένη μηχανή. Αυτή η ενδιάμεση αναπαράσταση πρέπει να έχει δυο σημαντικές ιδιότητες: πρέπει να είναι εύκολο να παραχθεί και

πρέπει να είναι εύκολο να μεταφραστεί στη γλώσσα της μηχανής στόχου. Στο μοντέλο ανάλυσης-σύνθεσης ενός μεταγλωττιστή , το εμπρός τμήμα αναλύει ένα πηγαίο πρόγραμμα (source program) και δημιουργεί μια ενδιάμεση αναπαράσταση, από την οποία το πίσω τμήμα παράγει τον τελικό κώδικα (target code). Ιδανικά, οι λεπτομέρειες την πηγαίας γλώσσας περιορίζονται στο εμπρός τμήμα και οι λεπτομέρειες της μηχανής στόχου στο πίσω τμήμα. Στη διαδικασία μετάφρασης ενός προγράμματος μιας δεδομένης πηγαίας γλώσσας σε κώδικα για μια δεδομένη μηχανή στόχο, ο μεταγλωττιστής κατασκευάζει μια ακολουθία ενδιάμεσων αναπαραστάσεων. Οι υψηλού επιπέδου αναπαραστάσεις είναι κοντά στην πηγαία γλώσσα και οι χαμηλού επιπέδου αναπαραστάσεις είναι κοντά στη μηχανή στόχο. Μια αναπαράσταση χαμηλού επιπέδου είναι κατάλληλη για εργασίες που εξαρτώνται από τη μηχανή όπως η κατανομή καταχωρητών και η επιλογή εντολών. Η σχεδίαση μιας ενδιάμεσης αναπαράστασης είναι η γλώσσα C. Η C είναι μια γλώσσα προγραμματισμού, όμως χρησιμοποιείται συχνά ως ενδιάμεση μορφή επειδή είναι ευέλικτη, μεταγλωττίζεται σε αποδοτικό κώδικα μηχανής και οι μεταγλωττιστές της είναι ευρέως διαθέσιμοι.

Πίνακας Συμβόλων

Μια βασική λειτουργία ενός μεταγλωττιστή είναι η καταγραφή των ονομάτων των μεταβλητών που χρησιμοποιούνται στο πηγαίο πρόγραμμα και η συλλογή πληροφοριών σχετικά με τις διάφορες ιδιότητες κάθε ονόματος. Αυτές οι ιδιότητες μπορεί να παρέχουν πληροφορίες σχετικά με τον αποθηκευτικό χώρο που που διατίθεται για ένα όνομα, τον τύπο του, την εμβέλεια (που μπορεί να χρησιμοποιηθεί η τιμή του μέσα στο πρόγραμμα) και στην περίπτωση των ονομάτων διαδικασιών, πληροφορίες όπως ο αριθμός και ο τύπος των ορισμάτων τους, η μέθοδος περάσματος κάθε ορίσματος (για παράδειγμα με τιμή ή αναφορά) και ο τύπος που επιστρέφεται. Ο πίνακας συμβόλων είναι μια δομή δεδομένων που περιέχει μια εγγραφή για κάθε όνομα μεταβλητής, με πεδία για τις ιδιότητες του ονόματος. Η δομή δεδομένων πρέπει να είναι κατάλληλα σχεδιασμένη ώστε να επιτρέπει στον μεταγλωττιστή να βρίσκει μια εγγραφή για ένα όνομα γρήγορα

και να αποθηκεύει ή να ανακτά δεδομένα από αυτή την εγγραφή για ένα όνομα γρήγορα και να αποθηκεύει ή να ανακτά δεδομένα από αυτή την εγγραφή επίσης γρήγορα. Οι εγγραφές του πίνακα συμβόλων δημιουργούνται και χρησιμοποιούνται κατά τη φάση της ανάλυσης από το λεκτικό αναλυτή, το συντακτικό αναλυτή και το σημασιολογικό αναλυτή.

Παραγωγή Τελικού Κώδικα

Η τελική φάση στο μοντέλο του μεταγλωττιστή μας είναι ο παράγωγος κώδικα. Δέχεται ως είσοδο την ενδιάμεση αναπαράσταση που παράγεται από το εμπρός τμήμα του μεταγλωττιστή, μαζί με τις σχετικές πληροφορίες του πίνακα συμβόλων και παράγει ως έξοδο ένα σημασιολογικά ισοδύναμο τελικό πρόγραμμα. Το τελικό πρόγραμμα διατηρεί το σημασιολογικό νόημα του πηγαίου προγράμματος και να είναι υψηλής ποιότητας, δηλαδή πρέπει να κάνει αποτελεσματική χρήση των διαθέσιμων πόρων της μηχανής στόχου. Επιπλέον, ο ίδιος ο παραγωγός κώδικα εκτελείται αποδοτικά. Ο παραγωγός κώδικα πραγματοποιεί τρεις βασικές εργασίες: την επιλογή εντολών, την κατανομή και την ανάθεση καταχωρητών και τη διάταξη των εντολών. Η επιλογή εντολών έχει να κάνει με την επιλογή κατάλληλων εντολών για την τελική μηχανή. Η κατανομή και ανάθεση καταχωρητών περιλαμβάνει την απόφαση του ποιες τιμές πρέπει να κρατηθούν και σε ποιους καταχωρητές.

Ανάλυση Συναρτήσεων Κώδικα

Συνάρτηση lex()

Η συνάρτηση lex ορίζει έναν λεκτικό αναλυτή, ορίζοντας κανονικές εκφράσεις που περιγράφουν πρότυπα για λεκτικές μονάδες. Παρασκηνιακά, η συνάρτηση lex μετασχηματίζει τα πρότυπα εισόδου σε ένα διάγραμμα μεταβάσεων και δίνει αποτελέσματα που προσομοιώνουν το διάγραμμα μεταβάσεων. Η lex διαβάζει αυτόματα ένα χαρακτήρα μπροστά από τον τελευταίο χαρακτήρα που σχηματίζει το επιλεγμένο λέξημα και κατόπιν ανακαλεί την είσοδο έτσι ώστε μόνο το ίδιο το λέξημα να αφαιρείται από την είσοδο. Η συνάρτηση lex μετατρέπει το πρόγραμμα εισόδου σε έναν λεκτικό αναλυτή, χρησιμοποιώντας ως κεντρική φιλοσοφία ένα πεπερασμένο αυτόματο καταστάσεων. Η lex χωρίζει την είσοδο σε λεξήματα και στη συνέχεια τα ελέγχει σαν πεπερασμένο αυτόματο μέχρι ο πίνακας μεταβάσεων να οδηγηθεί ή όχι σε μια τελική κατάσταση. Η συνάρτηση αναγνωρίζει τις δεσμευμένες λέξεις και τις δέχεται μόνο όταν πρέπει να τις δεχτεί. Τα πεπερασμένα αυτόματα είναι αναγνωριστές, δηλαδή λένε απλά “ναι”¨ή “όχι” για κάθε πιθανή συμβολοσειρά.

Συναρτήσεις Συντακτικής Ανάλυσης

Οι συναρτήσεις που χρησιμοποιούνται για τη συντακτική ανάλυση υπόκεινται στους κανόνες τις γραμματικής της γλώσσας Starlet. Στον πυρήνα της χρησιμοποιεί τη φιλοσοφία της Γραμματικής Χωρίς Συμφραζόμενα, με την κάθε μία συνάρτηση να καλεί κάποια άλλη ή και τον εαυτό της προκειμένου να φτάσει σε μία τελική κατάσταση σύμφωνα με τα αποτελέσματα του λεκτικού αναλυτή. Μια Γραμματική Χωρίς Συμφραζόμενα αποτελείται από τερματικά, μη-τερματικά, ένα αρχικό σύμβολο και παραγωγές. Τα τερματικά είναι τα βασικά σύμβολα από τα οποία σχηματίζονται οι συμβολοσειρές. Τα μη-τερματικά είναι συντακτικές μεταβλητές που υποδηλώνουν σύνολα από συμβολοσειρές. Τα σύνολα των συμβολοσειρών που υποδηλώνονται από μη-τερματικά βοηθούν στον ορισμό της γλώσσας που παράγεται από τη γραμματική. Τα μη-τερματικά επιβάλλουν μια ιεραρχική δομή στη γλώσσα που είναι το κλειδί για τη συντακτική ανάλυση και μετάφραση. Σε μία γραμματική, ένα μη-τερματικό χαρακτηρίζεται ως το αρχικό

σύμβολο και το σύνολο των συμβολοσειρών που υποδηλώνει, είναι η γλώσσα που παράγεται από τη γραμματική. Οι παραγωγές μιας γραμματικής καθορίζουν τον τρόπο με τον οποίο τα τερματικά και μη-τερματικά μπορούν να συνδυαστούν για να σχηματίσουν συμβολοσειρές.

Συναρτήσεις για την Παραγωγή Ενδιάμεσου Κώδικα

Η περιγραφή των εντολών καθορίζει τα συστατικά κάθε τύπου εντολής, αλλά δεν καθορίζει την αναπαράσταση αυτών των εντολών σε μια δομή δεδομένων. Στον μεταγλωττιστή, αυτές οι εντολές μπορούν να υλοποιηθούν ως αντικείμενα ή ως εγγραφές με πεδία για τον τελεστή και τους τελεστέους. Μια τέτοια αναπαράσταση είναι οι τετράδες τις οποίες χρησιμοποιούμε. Μία τετράδα έχει τέσσερα πεδία, τα οποία καλούμε op, arg_1 , arg_2 και result. Το πεδίο op περιέχει έναν εσωτερικό κωδικό για τον τελεστή. Για παράδειγμα, η εντολή τριών-διευθύνσεων $x = y + z$ αναπαρίσταται με την τοποθέτηση του + στο op, του y στο arg_1, του z στο arg_2 και του x στο result. Κάθε τετράδα έχει μπροστά της έναν μοναδικό αριθμό που την χαρακτηρίζει. Μόλις τελειώσει η εκτέλεση μια τετράδας εκτελείται η τετράδα που έχει τον αμέσως μεγαλύτερο αριθμό, εκτός και αν η τετράδα που μόλις εκτελέστηκε υποδείξει κάτι διαφορετικό.

Έτσι λοιπόν έχουμε:

- **nextquad()** η οποία επιστρέφει τον αριθμό της επόμενης τετράδας που πρόκειται να παραχθεί
- **genquad(op, x, y, z)** που δημιουργεί την επόμενη τετράδα *(op, x, y, z)*
- **newtemp()** η οποία δημιουργεί και επιστρέφει μια νέα προσωρινή μεταβλητή, της μορφής *T_1, T_2, T_3…..*
- **emptylist()** η οποία δημιουργεί μια κενή λίστα ετικετών τετράδων
- **makelist(x)** που δημιουργεί μια λίστα ετικετών τετράδων που περιέχει μόνο το *x*

- **merge(list$_1$, list$_2$)** που δημιουργεί μια λίστα ετικετών τετράδων από τη συνένωση των λιστών list$_1$, list$_2$
- **backpatch(list, z)** στην οποία 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 συμπληρώνει με την ετικέτα *z*.

Πίνακας Συμβόλων

Ο πίνακας συμβόλων κρατάει στοιχεία για τις συναρτήσεις του προγράμματος, τις μεταβλητές και τα πεδία. Παράλληλα για όλες τις μεταβλητές κρατάει την εμβέλεια τους μέσα στο πρόγραμμα. Ο όρος εμβέλεια από μόνος του αναφέρεται σε ένα τμήμα του προγράμματος που είναι η εμβέλεια μιας ή περισσότερων δηλώσεων. Οι εμβέλειες είναι σημαντικές, διότι ο ίδιος προσδιοριστής μπορεί να δηλωθεί για διαφορετικούς σκοπούς σε διαφορετικά μέρη του προγράμματος. Στην πράξη , ο ρόλος του πίνακα συμβόλων είναι να περνάει πληροφορίες από τις δηλώσεις στις χρήσεις. Η σημασιολογική δράση προσθέτει πληροφορίες σχετικά με έναν προσδιοριστή του πίνακα συμβόλων, όταν αναλύεται η δήλωσή του. Μεταγενέστερα, μια σημασιολογική δράση που συσχετίζεται με μια παραγωγή, παίρνει πληροφορίες για τον προσδιοριστή από τον πίνακα συμβόλων.

Παραγωγή Τελικού Κώδικα

Στην τελική αυτή φάση, της παραγωγής κώδικα, έχουμε φτάσει σε ένα σημείο που το πηγαίο πρόγραμμα έχει διερευνηθεί, αναλυθεί συντακτικά και μεταφραστεί σε μια σχετικά χαμηλού επιπέδου γλώσσα, έτσι ώστε οι τιμές των ονομάτων που εμφανίζονται να μπορούν να αναπαρασταθούν από ποσότητες τις οποίες μπορεί να χειριστεί άμεσα η τελική μηχανή. Επίσης, τα συντακτικά και στατικά σημασιολογικά σφάλματα έχουν ανιχνευτεί, έχει πραγματοποιηθεί ο απαραίτητος έλεγχος τύπων και οι τελεστές μετατροπής τύπων έχουν εισαχθεί οπουδήποτε αυτό είναι απαραίτητο. Ο παραγωγός επομένως προχωράει υποθέτοντας ότι η

είσοδος είναι απαλλαγμένη από αυτά τα είδη σφαλμάτων. Η αρχιτεκτονική του συνόλου εντολών της τελικής μηχανής είναι η RISC. Μία μηχανή RISC τυπικά έχει πολλούς καταχωρητές εντολών τριών-διευθύνσεων, απλούς τρόπους διευθυνσιοδότησης και μια σχετικά απλή αρχιτεκτονική συνόλου εντολών. Η κάθε εντολή του πηγαίου κώδικα μεταφράζεται σε κώδικα Assembly. Συγκεκριμένα:

- **gnlvcode(v)** η οποία αναζητά στον πίνακα συμβόλων την εμβέλεια της μεταβλητής v
- **loadvar(v, r)** η οποία σύμφωνα με την εμβέλεια της μεταβλητής εκτελεί τις εντολές φόρτωσης της Assembly χρησιμοποιώντας και προσωρινούς καταχωρητές.
- **storevar(r, v)** η οποία σύμφωνα με την εμβέλεια της μεταβλητής εκτελεί τις εντολές αποθήκευσης της Assembly.
- **transform_to_finalCode(quad)** η οποία δέχεται μια τετράδα και σύμφωνα με τις πληροφορίες που περιέχει επιστρέφει και την ανάλογη μετατροπή σε κώδικα Assembly.
- **generate_finalCode_file(finalCodeFile)** η οποία δέχεται ως όρισμα ένα αρχείο στο οποίο εξάγει το τελικό αποτέλεσμα.