ΑΝΑΦΟΡΑ ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗΣ ΑΣΚΗΣΗΣ

ΜΕΛΗ ΟΜΑΔΑΣ

ΚΩΝΣΤΑΝΤΙΝΟΣ ΜΟΥΡΟΥΣΙΔΗΣ 4114 cs04114

ΑΛΕΞΑΝΔΡΟΣ ΝΙΚΟΛΑΟΥ 4126 cs04126

Εισαγωγή

Στην προγραμματιστική αυτή άσκηση μας ζητήθηκε να υλοποιήσουμε έναν μεταφραστή για την γλώσσα C-Cimple. Ο μεταφραστής μας αποτελείται από τέσσερα τμήματα τον λεκτικό αναλυτή, το συντακτικό αναλυτή ,την παραγωγή του ενδιάμεσου κώδικα σε προγραμματιστική γλώσσα C και την παραγωγή του τελικού κώδικα σε προγραμματιστική γλώσσα assembnly. Παρακάτω αναλύονται σε ξεχωρly. Παρακάτω αναλύονται σε ξεχωριστά κεφάλαια τα παραπάνω τμήματα.

Κεφάλαιο 1ο

Λεκτικός αναλυτής

Περιγραφή

Ο λεκτικός αναλυτής καλείται ως συνάρτηση από το συντακτικό αναλυτή, διαβάζει γράμμα-γράμμα το αρχικό πρόγραμμα και κάθε φορά που καλείται επιστρέφει την επόμενη λεκτική μονάδα. Πρόκειται δηλαδή για ένα αυτόματο καταστάσεων που ξεκινά από μία αρχική κατάσταση, με την είσοδο κάθε χαρακτήρα αλλάζει κατάσταση έως ότου συναντήσει μία τελική κατάσταση.

Ανάλυση κώδικα

Η συνάρτηση που υλοποιεί τον λεκτικό αναλυτή στον κώδικα μας είναι η lex() (γραμμη:43).Αφού κληθεί η συνάρτηση αυτή από τον συντακτικό αναλυτη ,θετουμε μια μεταβλητη x μεσω της οποια διαβαζεται το προγραμμα χαρακτηρα-χαρακτηρα και με βαση το τι ακολουθει μετα από αυτόν τον χαρακτηρα σχηματιζονται ειτε λεξεις κλειδια ειτε ονοματα μεταβλητων ειτε αριθμοι ή συμβολα,τα οποια επιστρεφονται.Επισης μεσω των συνθηκων (γραμμες: 62, 93, 138, 144, 164, 184) ελεγχονται αντιστοιχα οι μη επιτρεπτες εκφρασεις στην γλωσσα C-Cimple,το μεγεθος του ονοματος της μεταβλητης να μην ξεπερνα τους 30 χαρακτηρες,να μην ακολουθει γραμμα μετα από αριθμο,μετα από το συμβολο “:” να ακολουθει το συμβολο “=”,μετα από το συμβολο “<” να ακολουθουν ή τα συμβολα “>”,”=” ή αριθμος ή γραμμα, μετα από το συμβολο “>” να ακολουθουν ή το συμβολο ”=” ή αριθμος ή γραμμα και υποχρεωτικα όταν ανοιγουν σχολια θα πρεπει να κλεινουν.

Κεφαλαιο 2ο

Συντακτικος αναλυτης

Περιγραφη

Ο συντακτικός αναλυτής καλείται αυτοματα στην αρχη του προγραμματος,για να ελέγξει εάν το πηγαίο πρόγραμμα ανήκει ή όχι στη γλώσσα,στη συνέχεια δημιουργεί το κατάλληλο περιβάλλον μέσα από το οποίο αργότερα θα κληθούν οι σημαντικές ρουτίνες. Για κάθε έναν από τους κανονες της γραμματιικης της γλώσσας , φτιάχνουμε και ένα αντίστοιχο υποπρόγραμμα. Οσο δεν συναναντται το τερματικο συμβολο καλειται το αντιστοιχο υποπρογραμμα. Όταν συναντηθει τερματικο συμβολο το τοτε αν και ο λεκτικος αναλυτης επιστρεψει λεκτική μονάδα που αντιστοιχεί στο τερματικό αυτό σύμβολο αυτό σημαινει ότι εχει αναγνωριστει επιτυχως η λεκτικη μοναδα.Αν όμως δεν επιστρεψει τη λεκτική μονάδα που περιμένει ο συντακτικός αναλυτής τοτε σημαινει ότι υπαρχει λαθος στον κωδικα και καλειται ο διαχειριστης σφαλματων και τυπωνει το αναλογο συνντακτικο λαθος. Όταν αναγνωριστεί και η τελευταία λέξη του κωδικα, τότε η συντακτική ανάλυση έχει πετυχει.

Ανάλυση κώδικα

Ο συντακτικός αναλυτής εκτείνεται από τη γραμμη 192 εως τη γραμμη 745.

Στη συναρτηση program() αναγνωριζεται το ονομα του προγραμματος και καλειται η συναρτηση block().Οταν επιστρεψει από την συναρτηση block() τοτε εχει τελειωσει η συντακτικη ανάλυση.

Στη συναρτηση block() , πρωτα καλειται η συναρτηση declarations() η οποια ελέγχει αν οριζονται σωστα οι μεταβλητες που υπαρχουν. Στη συνεχεια αν υπαρχουν υποπρογραμματα (procedures ή functions) στο πηγαιο προγραμμα καλειται η συναρτηση subprograms() αλλιως προχωραει στον ελεγχο της main του προγραμματος μεσω της statements().

Στη συναρτηση subprograms() καλειται η συναρτη subprogram(), η οποια καλεί ή την function() ή την procedure() αναλογα με το ειδος του υποπρογραμματος, οι δυο αυτες συναρτησεις εκτελουν παρομοια διαδικασια.Αρχικα ελεγχονται οι παραμετροι των υποπρογραμματων με την κληση formalparlist(),στην οποια επισης καλειται η formalparitem() για να καθορισει το ειδος των παραμετρων, και στη συνεχεια η statements() για τον ελεγχο του κωδικα του πηγαιου προγραμματος.

Στη συναρτηση statements() καλειται η συναρτηση statement() οσο υπαρχει πηγαιο προγραμμα.Στη συναρτηση statement() καλειται η συναρτηση στην οποια αντιστοιχει η λεξη κλειδι(whileStat(), ifStat(), switchcaseStat(), forcaseStat(), incaseStat(), callStat(), returnStat(), inputStat(), printStat()) ή το ονομα μεταβλητης(assignStat())

Η συναρτηση assignStat() καλεί την expression() για να καθορισει το ειδος της εκφρασης και ελεγχει ο ορισμος της.

Η συναρτηση ifStat() καλειται όταν εντοπιστεί η λεξη-κλειδι if και στη συνεχεια καλει την condition() για να ελεγξει την συνθηκη της if, καλει την statements() για τον ελεγχο του κωδικα της if και τελος

καλει την elsepart() για να ελεγξει μεσω της statements() τον κωδικα της εφοσον υπαρχει.

Η συναρτηση whileStat() καλειται όταν εντοπιστεί η λεξη-κλειδι while και στη συνεχεια καλει την condition() για να ελεγξει την συνθηκη της while, καλει την statements() για τον ελεγχο του κωδικα της while.

Η συναρτηση switchcaseStat() για κάθε περίπτωση (case) καλει την condition() για να ελεγξει την ορθοτητα της συνθηκης και την statements() για τον ελεγχο του κωδικα της.

Η συναρτηση forcaseStat() για κάθε περίπτωση (case) καλει την condition() για να ελεγξει την ορθοτητα της συνθηκης και την statements() για τον ελεγχο του κωδικα της.

Η συναρτηση incaseStat() για κάθε περίπτωση (case) καλει την condition() για να ελεγξει την ορθοτητα της συνθηκης και την statements() για τον ελεγχο του κωδικα της.

Η συναρτηση returnStat() καλει την expression() για να ελεγξει την εκφραση που επιστεφεται από την εντολη return.

Η συναρτηση callStat() καλει την ID() για να αναγνωρισει το ονομα του υποπρογραμματος και στη συνεχεια καλει την actualparlist() για να ελκεγξει το περασμα των παραμετρων.

Η συναρτηση printStat() καλει την expression() για να ελεγξει την εκφραση που θα τυπωθει.

Η συναρτηση inputStat() καλει την ID για να αναγνωρισει το ονομα της μεταβλητης που χρειαζεται.

Η συναρτηση condition() χρησιμοποιει την boolterm() για να ελεγξει την ορθοτητα των συνθηκων οσο υπαρχει το “or” ξανακαλειται. Αναλυτικοτερα η boolterm() καλει την boolfactor() και οσο υπαρχει το “and” ξανακαλειται. Η boolfactor() μεσω της expression() ελεγχει την εκφραση και όταν επιστρεφει αν εντοπιστει ένα από τα συμβολα ["=","<=",">=","<",">","<>"] ξανακαλειται η expression(), αν εντοπιστει “not” καλειται η condition() για ελεγχο της συνθηκης και το ιδιο ισχυει αν εντοπιστει “[“.

Η συναρτηση expression() καλει την term() και οσο υπαρχει “+,-” ξανακαλειται.Η term() καλει την factor() και οσο υπαρχει “*,/” ξανακαλειται και η factor() αν εντοπισει ειτε αριθμο ειτε ”(“ καλει την expression() και αν εντοπισει ονομα μεταβλητης τοτε καλει την function_id() ή την id_tail() αναλογα το ονομα.

Η συναρτηση funtion_id() ελεγχει τις παραμετρους μεσω της actualparlist().

Η συναρτηση idtail() ελεγχει τις παραμετρους μεσω της actualparlist() και στο τελος καλει την expression().

Η συναρτηση ID() καλειται κάθε φορα που χρεαιζετασι να γινει η αναγνωρισει καποιου ονοματος ειτε μεταβλητης ειτε υποπρογραμματος κτλ.

Κεφαλαιο 3ο

Παραγωγη Ενδιαμεσου Κωδικα

Περιγραφή

Η παραγωγη του ενδιαμεσου κωδικα υλοποιείται με τη δημιουργια ενός συνολου από τετραδες, οι οποιες αποτελουνται από έναν τελεστη και τρία τελουμενα. Οι τετράδες είναι αριθμημένες. Κάθε τετράδα έχει μπροστά της έναν μοναδικό αριθμό που τη χαρακτηρίζει. Μόλις τελειώσει η εκτέλεση μίας τετράδας εκτελείται η τετράδα που έχει τον αμέσως μεγαλύτερο αριθμό, εκτός εάν η τετράδα που μόλις εκτελέστηκε υποδείξει κάτι διαφορετικό . Με αυτόν τον τροπο κάθε τετραδα μπορει να μετατραπει σε μια εντολη της γλωσσας C.

Αναλυση Κωδικα

Ο κωδικας που υλοποιει την παραγωγη του ενδιαμεσου κωδικα συμπεριλαμβανεται στον κωδικα του συντακτικου αναλυτη και στη γραμμη 749 εως 836. Ουσιαστικα στις γραμμες αυτές εχουμε δημιουργησει συναρτησεις που βοηθουν να παραγουμε τον

ενδιαμεσο κωδικα.Οι συναρτησεις αυτές χρησιμοποιουνται και μεσα στον συντακτικο αναλυτη.

Η συναρτηση genquad() προσθετει μια τετραδα στον πινακα μας και αυξανει τον μετρητη για την επομενη θεση.

Η συναρτηση nextquad() επιστρεφει την επομενη θεση/γραμμη του πινακα μας.

Η συναρτηση newtemp() επιστρεφει μια προσωρινη μεταβλητη (τυπου : T_0).

Η συναρτηση emptylist() επιστρεφει έναν κενο πινακα.

Η συναρτηση makelist(x) παιρνει ως παραμετρο έναν αριθμο και επιστρεφει μια τετραδα οπου στη τελευταια θεση είναι αυτος ο αριθμος.

Η συναρτηση mergelist(list1,list2) παιρνει τις λιστες των παραμετρων και της συγχωνευει σε μια.

Η συναρτηση backpatch(list1,z) πρεπει ως παραμετρο μια λιστα και έναν αριθμο(z) και δημιουργει μια λιστα με μια τετραδα με τον αριθμο(z) αυτό στην τελευταια θεση .

Η συναρτηση printArray() τυπωνει τον πινακα με τις τετραδες.

Η συναρτηση transfer() γραφει σε ένα αρχειο τις τετραδες του πινακα.

Εφοσον εχει ολοκληρωθει η δημιουργια των τετραδων καλειται η συναρτηση convert_c() η οποια δημιουργει ένα αρχειο κωδικα C. Η συναρτηση αυτή αρχικα διαβαζει τα στοιχεια του πινακα και αποθηκευει ολες τις μεταβλητες και τις γραφει στο αρχειο του κωδικα C. Στη συνεχεια ξαναδιαβαζονται τα στοιχεια του κωδικα και αναλογα τις τετραδες,γραφονται οι αναλογες εντολές στο αρχειο του κωδικα C.

Κεφαλαιο 4ο

Παραγωγη Τελικου Κωδικα

Περιγραφή

Στην παραγωγη του τελικου κωδικα ,οι εντολες του ενδιαμεσου κωδικα μεσω τον τετραδων του πινακα παραγουν τις αντίστοιχες εντολές του τελικού κώδικα.

Αναλυση Κωδικα

Ο κωδικας που παραγει τον τελικο κωδικα εκτεινεται από τη γραμμη 838 εως τη γραμμη 930.Αυτο επιτυγχανεται με την κληση της συναρτησης telikos().

Η συναρτηση telikos() δημιουργει έναν αρχειο .asm που θα γραφτουν οι εντολες του τελικου κωδικα.Οπως και στον ενδιαμεσο ετσι και στον τελικο διαβαζουμε μια μια τις τετραδες του πινακα και αναλογα της τετραδας γραφονται στο αρχειο .asm οι καταλληλες εντολες στην γλωσσα assemply.

Στις εντολες αλματων:

- **jump, “_”, “_”, label**

  b label

- **relop(?),x,y,z**

  loadvr(x,$t1)

  loadvr(y, $t2)

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

Στην εκχωρηση:

- **:=, x, “_”, z**

  loadvr(x, $t1)

  storerv($t1, z)

Στις εντολες αριθμητικων πραξεων:

- **op x,y,z**

  loadvr(x, $t1)

  loadvr(y, $t2)

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

  storerv($t1,z)

Στις εντολες εισοδου-εξοδου:

- **out “_”, “_”, x**

  li $v0,1

  loadvr(x,$a0)

  syscall

- **in “_”, “_”, x**

  li $v0,5

  syscall

  storerv($v0,x)

Στην επιστροφη τιμης:

- **retv "_", "_", x**

  loadvr(x, $t1)

  lw $t0,-8($sp)

  sw $t1,($t0)

Στις παραμετρους συναρτησεις αναλογα τον αριθμος τις παραμετρου χρησιμοποιειται και το καταλληλο offset.

- **par,x,CV, _**

  loadvr(x, $t0)

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

  όπου i ο αύξων αριθμός της παραμέτρου

- **par,x,REF, _**

  addi $t0,$sp,-offset

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

- **par,x,RET, _**

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

Στην κληση συναρτησης:

- **call, _, _, f**

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

addi $sp,$sp,framelength

jal f

addi $sp,$sp,-framelength
```

Στην αρχη ενός μπλοκ υποπρογραμματος γραφουμε:

sw $ra,-0($sp)

Και στο τελος:

lw $ra,-0($sp)

jr $ra

Ενώ στην αρχη του πηγαιου προγραμματος γραφουμε:

```
j Lmain
```

Οπου Lmain το ονομα του πηγαιου προγραμματος.

```
addi $sp,$sp,framelength

   move $s0,$sp
```

Και στο τελος:

```
addi $sp,$sp,framelength

   move $s0,$sp
```

ΕΠΙΛΟΓΟΣ

Λειτουργικοτητα μεταφραστη

Οσο αφορα την λειτουργικοτητα του μεταφραστη μας υπαρχουν καποια προβληματα στην παραγωγη του ενδιαμεσου κωδικα και στην παραγωγη του τελικου.Ο λεκτικος αναλυτης και ο συντακτικος αναλυτης λειτουργει κανονικα και εχει ελεγθει με διαφορα προγραμματα στην γλωσσα C-Cimple.Απο την άλλη η παραγωγη ενδιαμεσου κωδικα εχει αρκετα προβληματα όπως δεν δημιουργει σωστα του πινακες με τα αλματα για το που θα πρεπει να κανει jump μια συνθηκη και αυτό γινεται όταν σε καποιο if ή while υπαρχουν παραπανω από μια συνθηκες με “or” ή “and”.Ακομη ένα προβλημα είναι επισης στα αλματα τις switchcase.Επιπλεον για την

παραγωγη τελικου κωδικα δεν εχουμε υλοποιησει τον πινακα συμβολων με αποτελεσμα ο τελικος κωδικας να είναι ημιτελης.

Ευχαριστούμε για τον χρόνο σας,

οι φοιτητές

Κωνσταντίνος Μουρουσίδης ,

Αλέξανδρος Νικολάου!