Εξαμηνιαία Εργασία
Μεταγλωττιστές 2017

Στρατή Φωτεινή
Αναγνωστίδης Σωτήρης-Κωνσταντίνος

Το παρόν αποτελεί μια σύντομη αναφορά της εξαμηνιαίας εργασίας στο μάθημα μεταγλωττιστές 2017 που αφορά τη γλώσσα προγραμματισμού dana. Παρακάτω γίνεται μια σύντομη αναφορά των επιμέρους σημείων καθώς και ένα σύνολο παραδειγμάτων

Χρησιμοποιήθηκαν τα εργαλεία flex για την υλοποίηση του λεκτικού αναλυτή, bison για την υλοποίηση του συντακτικού αναλυτή, ενώ η παραγωγή τελικού κώδικα γίνεται μέσω llvm.

Λεκτικός Αναλυτής

Στα πλαίσια του πρώτου κομματιού της εργασίας, υλοποιήσαμε το λεκτικό αναλυτή. Κατά την υλοποίηση αυτή λήφθηκε υπόψη και ο offside rule. Δεν απαιτείται κάποια τροποποίηση από την πλευρά του χρήστη και είναι δυνατή η ταυτόχρονη χρήση “begin”, “end” για τη δεικτοδότηση ενός block, όσο και απλά η χρήση κενών και tab. Το γεγονός επιτεύχθηκε με το να εισάγουμε “begin”, “end” όπου αυτά λείπουν πριν το πέρασμα στο συντακτικό αναλυτή.

Συντακτικός Αναλυτής

Πριν το διάβασμα του αρχείου επιβάλλουμε την εισαγωγή ενός αρχείου που περιέχει τις επικεφαλίδες των συναρτήσεων βιβλιοθήκης που χρησιμοποιούντα. Στην αρχική αυτή μορφή επιτρέπουμε μόνο ένα περιορισμένο αριθμό συναρτήσεων, ωστόσο ο αριθμός αυτός μπορεί εύκολα να μεταβληθεί. Για το σκοπό αυτό τροποποιούμε ελαφρά τη γραμματική, επιτρέποντας δηλώσεις συναρτήσεων αρχικά και πριν τη συνάρτηση main. Το γεγονός αυτό είναι απαραίτητο για να εξασφαλιστεί σωστή χρήση της υπογραφής της κάθε συνάρτησης, έλεγχος που λαμβάνει χώρα στη συνέχεια και κατά τη διάρκεια της σημασιολογικής ανάλυσης.

Σημασιολογική Ανάλυση

Κατά τη διάρκεια της ανάλυσης αυτής, πραγματοποιούνται όλοι οι απαραίτητοι έλεγχοι για την εξασφάλιση σωστής λειτουργίας του προγράμματος. Τονίζουμε ότι εφαρμόσαμε αυστηρό έλεγχο τύπων, με πράξεις μεταξύ τύπων byte και int να μην επιτρέπονται. Όλοι οι σημασιολογικοί έλεγχοι γίνονται ταυτόχρονα με την κατασκευή του llvm flow. Ο μόνος έλεγχος που δεν πραγματοποιήθηκε αφορά το μη πέρασμα διαστάσεων πίνακα, που ωστόσο δεν αποτελεί πρωτόγονη προϋπόθεση.

Τέλος, για να είναι δυνατή η αναγνώριση μεταβλητών εμφωλευμένων συναρτήσεων καταφύγαμε στην χρήση global μεταβλητών (μετά από άπειρες ώρες ψάξιμο…). Θεωρούμε ότι η αλλαγή αυτή αντιβαίνει ελαφρώς σε πιθανή πραγματική εφαρμογή του συγκεκριμένου μεταγλωττιστή, αλλά είναι σε θεμιτά πλαίσια αλλαγής εκ μέρους μας.

Ακολουθούν μια σειρά παραδειγμάτων που αξιολογούν τα διάφορα ενδεχόμενα που μπορούν τα προκύψουν.

Λεκτική ανάλυση:

```
def main
  var c $ is byte
  decl add is int
```

```
test:2:  Error : symbol '$' not recognized
```

```
def main
  var c  is byte
  (* this is a comment (* inside a comment *)
```

```
test:4:  Error : Comment never closed
```

Συντακτική ανάλυση:

```
def main
  var c # is byte
  decl add is int
```

```
test:3:  Error : Could not find appropriate syntax, last read token decl
```

Σημασιολογική ανάλυση:

```
def main
  var c is byte
  var c is int
```

```
test:3:  Error : Duplicate identifier: c
```

```
def main
  var c is byte
  c[2] := 3
```

```
test:3:  Error : Variable  c is not an array
```

```
def main
  var c is byte[10]
  c := 3
```

```
test:3:  Error : wrong lvalue: c
```

```
def main
  var c is byte[10]
  loop a:
        loop a:
                break
```

```
test:4:  Error : Duplicate loop identifier a
```

```
def main
        var a is byte[10]
        def add is int: c as int
                c := 4
                return: c
        writeInteger: add(a)
```

```
test:6:  Error : type mismatch in parameter of the calling function: add
```

```
def main
        var c is byte[10]
        var i is int
        i := 0
        loop:
                c[i] := 'b'
                i := i + 1
                if i > 9:
                        break
        i := 0
        loop a:
                loop b:
                        if i > 4:
                                break: a
                        c[i] := 'a'
                        i := i + 1
        writeString: c
        writeString: "\n"
```

```
sotiris (master *) llvm_test $ ./test.out
aaaaabbbbb
```

```
def main
        var c is byte[10]
        def add is int: c as int
                c := c * 4
                return: c
        writeInteger: add(2)
        writeString: "\n"
```

```
sotiris (master *) llvm_test $ ./test.out
8
```