Απαλλακτική Εργασία στη Σχεδίαση Γλωσσών Προγραμματισμού

Γραμμένος Θεόδωρος

Α.Ε.Μ.: 3294

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Στο έγγραφο αυτό περιγράφεται η δική μου υλοποίηση της απαλλακτικής εργασίας στο μάθημα της σχεδίασης γλωσσών προγραμματισμού. Ειδικότερα, θα αναλυθεί ο τρόπος λειτουργίας του μεταγλωττιστή και θα περιγραφούν τα σημεία τα οποία είναι απαραίτητο. Παραδείγματα προγραμμάτων στη γλώσσα της εργασίας περιλαμβάνονται στο φάκελο Examples.

1.1 Τεχνικές Λεπτομέρειες

Ο μεταγλωττιστής υλοποιήθηκε στη γλώσσα Kotlin. Για τη διαδικασία της λεκτικής και της συντακτικής ανάλυσης χρησιμοποιήθηκε το εργαλείο ANTLR, το οποίο δημιουργεί καθοδικούς αναλυτές. Ο μεταγλωττιστής παράγει ένα αρχείο το οποίο περιέχει τον κώδικα σε MIXAL για το συγκεκριμένο πρόγραμμα. Έπειτα το αρχείο αυτό μπορεί να γίνει compile και να εκτελεστεί από έναν εξομοιωτή της μηχανής MIX. Κατά τις δοκιμές χρησιμοποιήθηκε ο εξομοιωτής MIX Builder. Τα αρχεία του λεκτικού και συντακτικού αναλυτή βρίσκονται στον φάκελο src/main/antlr.

Κεφάλαιο 2

Τρόπος Λειτουργίας

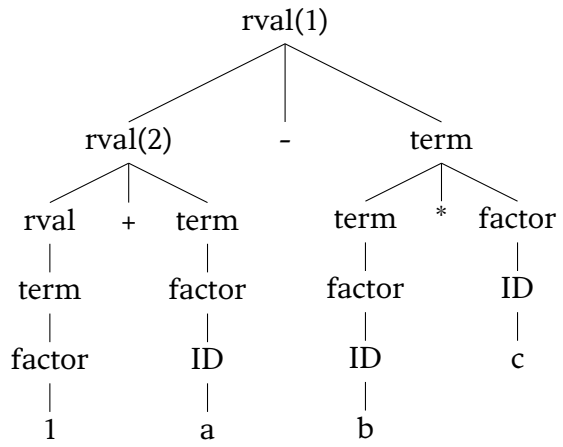
2.1 Γενικές πληροφορίες

Η βασική κλάση του μεταγλωττιστή είναι η ProjectListener. Η συγκεκριμένη κλάση υλοποιεί το interface ProjectParserBaseListener το οποίο δημιουργεί αυτόματα το ANTLR. Το inteface περιέχει μεθόδους enter και exit για κάθε σύμβολο της γραμματικής. Οι μέθοδοι αυτοί καλούνται από τον parser καθώς δημιουργείται το συντακτικό δέντρο του αρχείου εισόδου. Η μέθοδος enter ενός συμβόλου καλείται όταν το σύμβολο αναγνωρίζεται αρχικά, και δεν έχουν αναγνωριστεί τα παιδιά του. Κάτι τέτοιο είναι χρήσιμο όταν π.χ. δημιουργείται ένα καινούριο block και χρειάζεται να δημιουργηθεί πίνακας συμβόλων για το block. Αντίστοιχα, η μέθοδος exit καλείται όταν έχει ολοκληρωθεί η επεξεργασία του υποδένδρου με ρίζα αυτό το σύμβολο.

Το κάθε σύμβολο της γραμματικής περιέχει ένα πεδίο info τύπου ExprInfo το συγκεκριμένο πεδίο περιλαμβάνει πληροφορίες για το κάθε σύμβολο, όπως τη θέση του στη μνήμη, την τιμή του και τις οδηγίες που χρειάζονται για να υπολογιστεί. Για παράδειγμα, το πεδίο info μιας μεταβλητής περιέχει την θέση της μεταβλητής στην μνήμη και τον τύπο της, ενώ μιας αριθμητικής πράξης περιέχει τις οδηγίες που χρειάζεται για να υπολογιστεί και τη θέση στη μνήμη που αποθηκεύει το αποτέλεσμά της.

Ο μεταγλωτιστής αξιοποιεί κυρίως τις μεθόδους exit. Οι μέθοδοι enter χρησιμοποιούνται για την αλλαγή της κατάστασης του μεταγλωττιστή. Καθώς ολοκληρώνεται η επεξεργασία των συμβόλων καλούνται διαδοχικά οι μέθοδοι exit για τα σύμβολα από κάτω προς τα πάνω. Ταυτόχρονα προστίθονται οι απαραίτητες πληροφορίες στο πεδίο info, καθώς και οι απαραίτητες οδηγίες Assembly. Παρακάτω φαίνεται το συντακτικό δέντρο μιας έκφρασης στη γλώσσα.

Στο παραπάνω παράδειγμα: Αρχικά προστίθονται πληροφορίες σχετικά



Σχήμα 2.1: Συντακτικό δέντρο της έκφρασης 1+a-b*c

με τους τελεστέους. Για το 1 αποθηκεύεται ότι είναι literal αριθμός και η τιμή του, ενώ για το a και το b ελέγχεται εάν έχουν δηλωθεί, αποθηκεύεται ο τύπος τους και η θέση τους στη μνήμη. Στη συνέχεια, καθώς "ανεβαίνουμε" στο δέντρο στο rval(2) ελέγχεται ο τύπος των μεταβλητών και τοποθετούνται οι οδηγίες για τον υπολογισμό της έκφρασης στο πεδίο info. Επίσης στο πεδίο info αποθηκεύεται ότι π.χ. ότι το αποτέλεσμα βρίσκεται στον καταχωρητή A και είναι τύπου float. Ομοίως υπολογίζονται οι οδηγίες για την έκφραση b*c. Στο rval(1) τα πράγματα είναι πιο σύνθετα καθώς χρειάζεται πρώτα να υπολογιστούν οι εκφράσεις από κάτω. Λόγω της προτεραιότητας των πράξεων, τοποθετούνται πρώτα οι απαραίτητες οδηγίες για τον υπολογισμό του b*c. Στη συνέχεια το αποτέλεσμα αποθηκεύεται στη στοίβα προκειμένου να υπολογιστεί η έκφραση 1+a. Τέλος, ανακαλείται το αποτέλεσμα από τη στοίβα και υπολογίζεται η τελική τιμή της έκφρασης.

Το συγκεκριμένο παράδειγμα είναι αρκετά απλό. Στην πραγματικότητα ενδεχομένως να χρειάζονται μετατροπές μεταβλητών και σταθερών από int σε float, κάτι που περιπλέκει το πρόγραμμα.

2.2 Δομές επιλογής και επανάληψης

Η υλοποίηση των δομών επιλογής και επανάληψης είναι πιο σύνθετη καθώς εμπλέκονται μεταπηδήσεις από το ένα σημείο του κώδικα σε άλλο.

Δομή επανάληψης

Η γλώσσα υποστηρίζει τις εντολές while και for. Θα αναλυθεί η δομή της while καθώς μια for μπορεί πολύ εύκολα να μετατραπεί σε while.

Η γενική μορφή ενός while loop σε κώδικα Assembly είναι:

```
LDA ...   ⎫
ADD ...   ⎪
.........  } Υπολογισμός της συνθήκης
.........  ⎪
CMPA ...  ⎭
Jxx *+... Έλεγχος της συνθήκης
LDA ...   ⎫
.......    } Εντολές μέσα στην επανάληψη
.......   ⎭
JMP *-... Επιστροφή στην αρχή
```

Σχήμα 2.2: Γενική μορφή επανάληψης

Όπως φαίνεται παραπάνω πρέπει πρώτα να είναι γνωστός ο αριθμός των εντολών για το κάθε μέρος έτσι ώστε να υπολογιστούν τα ακριβή νούμερα για τις εντολές JUMP. Συνεπώς, όταν γίνεται είσοδος σε ένα loop μέσω της αντίστοιχης συνάρτησης enter σταματάει το output στο αρχείο, υπολογίζονται οι εντολές που χρειάζεται και αντί να γίνεται output σε αρχείο τοποθετούνται στο πεδίο info. Φτάνοντας στο σύμβολο του loop με τη μέθοδο exit, έχουν υπολογιστεί οι εντολές για όλα τα παιδιά και συνεπώς μπορούν να υπολογιστούν οι τιμές των JUMP και να δομηθεί το loop.

Σε περίπτωση που το loop δεν είναι εμφολευμένο οι εντολές γράφονται στο αρχείο, ενώ αν είναι τοποθετούνται στο πεδίο info για να χρησιμοποιηθούν από το εξωτερικό loop.

Δομή επιλογής

Στη δομή επιλογής συναντώνται παρόμοιες δυσκολίες με την εντολή JUMP και αντιμετωπίζονται όπως και στη δομή επανάληψης.

Η δομή επιλογής αποτελείται από το if-part και, προαιρετικά, από το else-part. Οπότε, απαιτείται τουλάχιστον μία εντολή JUMP.

- Όταν δεν υπάρχει else-part η εντολή απλώς παραλείπει το if.
- Όταν υπάρχει else-part το πρώτο jump μεταβαίνει στο else-part και τοποθετείται ένα επιπλέον JUMP στο τέλος του if-part το οποίο παραλείπει το else-part.

2.3 Εκτύπωση αριθμών float

Η γλώσσα του υπολογιστή δεν παρέχει κάποια έτοιμη εντολή για την εκτύπωση αριθμών τύπου float. Παρέχεται η εντολή FIX η οποία μετατρέπει έναν πραγματικό σε ακέραιο με στρογγυλοποίηση. Για αυτόν τον λόγο για να εκτυπωθεί ένας float αριθμός ακολουθούνται τα παρακάτω βήματα:

1. Αν ο αριθμός είναι θετικός αφαίρεσε 0.499, αν είναι αρνητικός πρόσθεσε 0.499. Ο αριθμός είναι 0.499 και όχι 0.5 για να αποφευχθούν σφάλματα που οφείλονται στην ακρίβεια των float αριθμών.

2. Εκτέλεσε FIX, εκτύπωσε το ακέραιο μέρος.

3. Αφαίρεσε το ακέραιο μέρος από τον αριθμό.

4. Πολλαπλασσίασε με το 10.000.

5. Κάνε FIX.

6. Εκτύπωσε τον αριθμό που προκύπτει, δηλαδή το δεκαδικό μέρος.

Ως αποτέλεσμα η εκτύπωσει ενός float αριθμού απαιτεί περισσότερες εντολές σε σχέση με την εκτύπωση ενός integer.

Κεφάλαιο 3

Περιγραφή των κλάσεων

exprInfo

Περιέχει πληροφορίες για ένα σύμβολο της γραμματικής της γλώσσας. Το πεδίο location μπορεί να περιέχει 3 ειδικές τιμές:

- LOCATION_A όταν το αποτέλεσμα της έκφρασης βρίσκεται στον καταχωρητή A.
- LOCATION_TEMP όταν το αποτέλεσμα βρίσκεται στην θέση TEMP.
- LITERAL όταν η έκφραση αποτελεί αριθμητικό literal μέσα στο αρχείο.

AssemblyWriter

Χειρίζεται τη δόμηση του αρχείου και την εγγραφή των εντολών σε αυτό. Επίσης, παρέχει στατικες μεθόδους για το formatting των εντολών.

ErrorWriter

Παρέχει μεθόδους για την προβολή σφαλμάτων και προειδοποιήσεων κατά τη διάρκεια της μεταγλώττισης.

InstructionList

Λίστα με String, τα οποία είναι σωστά διαμορφωμένα ως οδηγίες της MIX. Παρέχει μεθόδους για την τοποθέτηση και διαμόρφωση εντολών της MIX.

SymRec

Περιέχει πληροφορίες για ένα σύμβολο της γλώσσας.

SymTable

Αντιπροσωπεύει τον πίνακα συμβόλων για ένα scope της γλώσσας. Περιέχει ένα HashTable στο οποίο αποθηκεύονται τα σύμβολα. Παρέχει μεθόδους για τοποθέτηση και ανάκτηση συμβόλων.

SymTableManager

Διαχειρίζεται πολλαπλούς πίνακες συμβόλων. Περιέχει μεθόδους για την δημιουργία νέων πινάκων και τη διαγραφή υπάρχοντων. Οι πίνακες αποθηκεύονται σε μία στοίβα, με τους νεότερους να τοποθετούνται στην κορυφή.

VarType

Enum με τους διαθέσιμους τύπους μεταβλητών

Mover

Περιέχει μεθόδους οι οποίες δημιουργούν τις απαραίτητες εντολές για τον χειρισμό μεταβλητών στη μνήμη. Για παράδειγμα η μέθοδος moveToA δέχεται ένα αντικείμενο τύπου ExprInfo και επιστρέφει τις οδηγίες που χρειάζονται για να μεταφερθέι το σύμβολο στον καταχωρητή A.

Translator

Χειρίζεται την μετατροπή των boolean operators σε κατάλληλες εντολές JUMP.

Κεφάλαιο 4

Παραδείγματα

Σε αυτήν την ενότητα θα εξεταστεί η μεταγλώττιση των παραδειγμάτων που παρέχονται στην εκφώνηση της εργασίας.

4.1 Fibonacci

Το πρόγραμμα υπολογίζει τους πρώτους 10 αριθμούς της σειράς Fibonacci. Για λόγους πληρότητας παρατίθεται ο κώδικας.

```
1  mainclass Fibonacci {
2  public static void main ( ){
3      int first, second, i, tmp;
4      first=0;
5      second=1;
6      i=0;
7      while (i<10){
8          i=i+1;
9          tmp=first+second;
10         println (tmp);
11         first=second;
12         second=tmp;
13         }
14     }
15 }
```

Παρακάτω βρίσκεται ο κώδικας που παράγεται από τον μεταγλωττιστή.

```
1              ORIG 1
2  TERM        EQU 18
3              ORIG 5
4  TEMP        CON 0
5  TEMPF       CON 0
6  TOP         CON 0
7  STACK       ORIG *+50
```

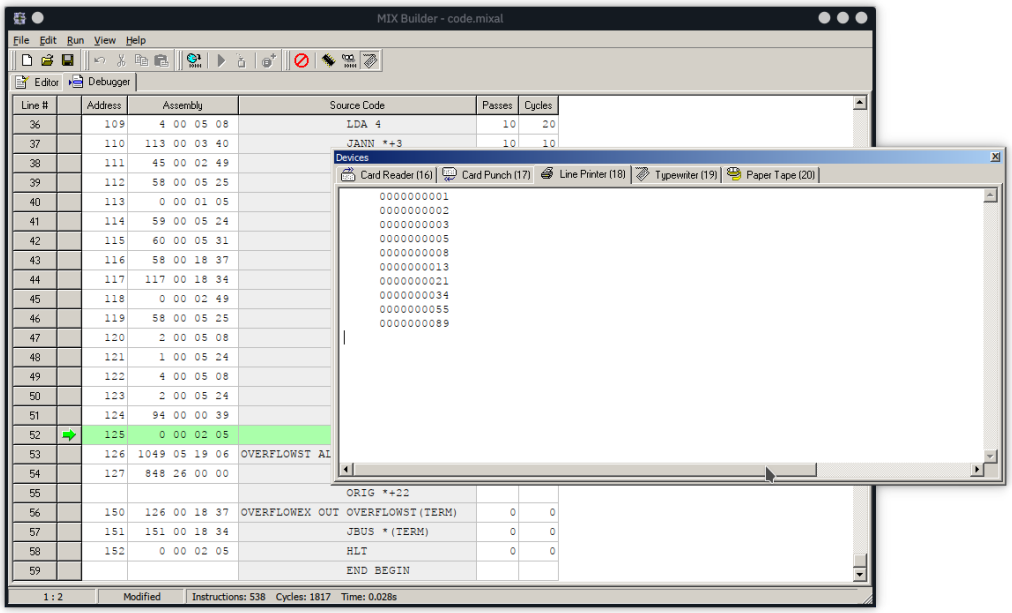
```
 8  LINE          ORIG  *+24
 9  BEGIN         NOP
10                ENTA  0
11                STA   1
12                STA   2
13                STA   3
14                STA   4
15                ENTA  0
16                STA   1
17                ENTA  1
18                STA   2
19                ENTA  0
20                STA   3
21                LDA   3
22                ENTX  10
23                STX   TEMP
24                CMPA  TEMP
25                JGE   *+27
26                LDA   3
27                ENTX  1
28                STX   TEMP
29                ADD   TEMP
30                JOV   OVERFLOWEX
31                STA   3
32                LDA   1
33                ADD   2
34                JOV   OVERFLOWEX
35                STA   4
36                LDA   4
37                JANN  *+3
38                ENT1  45
39                ST1   LINE
40                CHAR
41                STA   LINE+1
42                STX   LINE+2
43                OUT   LINE(TERM)
44                JBUS  *(TERM)
45                ENT1  0
46                ST1   LINE
47                LDA   2
48                STA   1
49                LDA   4
50                STA   2
51                JMP   *-30
52                HLT
53  OVERFLOWST    ALF   OVERF
54                ALF   LOW
55                ORIG  *+22
56  OVERFLOWEX    OUT   OVERFLOWST(TERM)
```

```
57                  JBUS  *(TERM)
58                  HLT
59                  END  BEGIN
```

Οι γραμμές 1-9 και 53-59 είναι σχεδόν ίδιες για όλα τα προγράμματα. Η μόνη διαφορά βρίσκεται στην εντολή ORIG στη γραμμή (3), η οποία διαφοροποιείται ανάλογα με τον αριθμό των μεταβλητών που περιέχει το πρόγραμμα. Οι γραμμές 53-59 περιέχουν οδηγίες για να εμφανιστεί μήνυμα όταν προκύπτει κάποια εξαίρεση.

Η δήλωση των μεταβλητών γίνεται στις γραμμές 10-14 του παραγόμενου κώδικα. Εισάγεται 0 στον καταχωρητή A και και αποθηκεύεται στη θέση μνήμης κάθε μεταβλητής. Οι αναθέσεις στις γραμμές 4-6 του αρχικού κώδικα πραγματοποιούνται στις γραμμές 15-20 του κώδικα MIX. Στις γραμμές 21-25 γίνεται ο έλεγχος της συνθήκης της επανάληψης. Στον καταχωρητή A φορτώνεται η μεταβλητή i, στη θέση TEMP το 10 και γίνεται η σύγκριση. Σε περίπτωση που δεν ισχύει η συνθήκη γίνεται JUMP 27 γραμμές μπροστά, δηλαδή στη γραμμή 52. Όταν ικανοποιείται η συνθήκη της επανάληψης, η εκτέλεση συνεχίζεται κανονικά. Στις γραμμές 26-50 εκτελούνται οι εντολές μέσα στην επανάληψη. Στη γραμμή 51 υπάρχει μια εντολή JUMP προς τη γραμμή 21, στον έλεγχο της συνθήκης της επανάληψης.

Παρακάτω φαίνεται ένα screenshot από την εκτέλεση του προγράμματος στο MIXBuilder.



Σχήμα 4.1: Τα αποτελέσματα εκτέλεσης του 1ου παραδείγματος.

Επίσης, στον ακόλουθο πίνακα φαίνεται η αντιστοιχία των γραμμών του αρχικού κώδικα με τον παραγόμενο.

Αρχικός Κώδικας	Κώδικας MIX
3	10-14
4	15-16
5	17-18
6	19-20
7	21-25,51
8	26-31
9	32-35
10	36-46
11	47-48
12	49-50

Πίνακας 4.1: Αντιστοιχία γραμμών

4.2 Μέσος όρος

Παρακάτω δίνονται ο αρχικός και ο παραγόμενος κώδικας για το δεύτερο παράδειγμα.

```
1      mainclass Example {
2  public static void main ( ){
3      int c;
4      float x, sum, mo;
5      c=0;
6      x=3.5;
7      sum=0.0;
8      while (c<5){
9          sum=sum+x;
10         c=c+1;
11         x=x+1.5;
12     }
13     mo=sum/5;
14     println (mo);
15     }
16 }
```

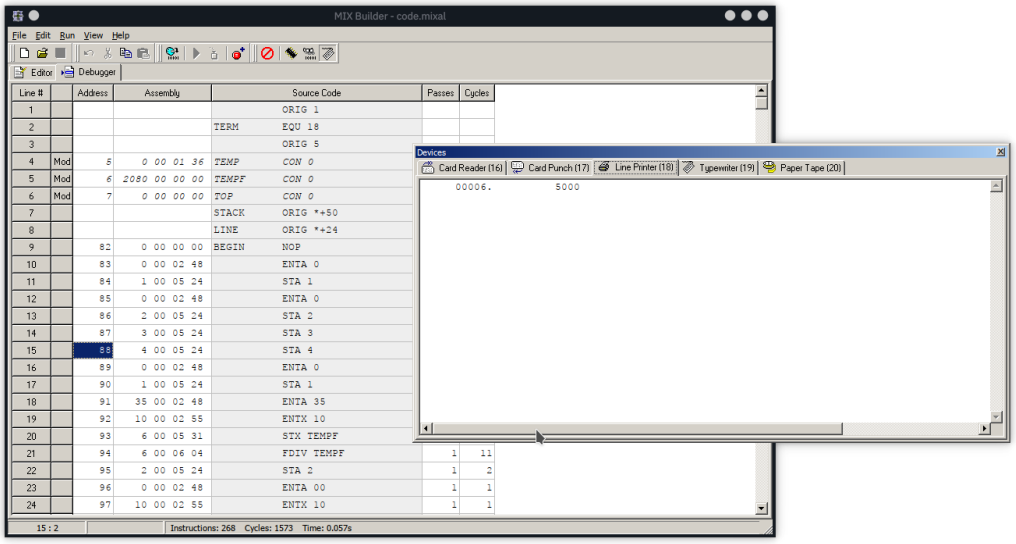
```
1               ORIG 1
2  TERM         EQU 18
3               ORIG 5
4  TEMP         CON 0
5  TEMPF        CON 0
```

```
 6  TOP         CON 0
 7  STACK       ORIG *+50
 8  LINE        ORIG *+24
 9  BEGIN       NOP
10              ENTA 0
11              STA 1
12              ENTA 0
13              STA 2
14              STA 3
15              STA 4
16              ENTA 0
17              STA 1
18              ENTA 35
19              ENTX 10
20              STX TEMPF
21              FDIV TEMPF
22              STA 2
23              ENTA 00
24              ENTX 10
25              STX TEMPF
26              FDIV TEMPF
27              STA 3
28              LDA 1
29              ENTX 5
30              STX TEMP
31              CMPA TEMP
32              JGE *+29
33              LDA 3
34              FADD 2
35              JOV OVERFLOWEX
36              STA 3
37              LDA 1
38              ENTX 1
39              STX TEMP
40              ADD TEMP
41              JOV OVERFLOWEX
42              STA 1
43              LDA 2
44              LD5 TOP
45              STA STACK,5
46              INC5 1
47              ST5 TOP
48              ENTA 15
49              ENTX 10
50              STX TEMPF
51              FDIV TEMPF
52              STA TEMP
53              LD5 TOP
54              DEC5 1
```

```
 55              LDA  STACK,5
 56              ST5  TOP
 57              FADD TEMP
 58              JOV  OVERFLOWEX
 59              STA  2
 60              JMP  *-32
 61              LDA  3
 62              LD5  TOP
 63              STA  STACK,5
 64              INC5 1
 65              ST5  TOP
 66              ENTX 5
 67              STX  TEMP
 68              LDA  TEMP
 69              FLOT
 70              STA  TEMP
 71              LD5  TOP
 72              DEC5 1
 73              LDA  STACK,5
 74              ST5  TOP
 75              FDIV TEMP
 76              JOV  OVERFLOWEX
 77              STA  4
 78              LDA  4
 79              STA  TEMP
 80              ENTA 1000
 81              STA  TEMPF
 82              ENTA 499
 83              FDIV TEMPF
 84              STA  TEMPF
 85              LDA  TEMP
 86              JAN  *+3
 87              FSUB TEMPF
 88              JMP  *+2
 89              ENT1 45
 90              ST1  LINE
 91              STA  TEMPF
 92              CHAR
 93              STX  LINE+1
 94              LDA  TEMPF
 95              FLOT
 96              STA  TEMPF
 97              ENTA 40
 98              SLA  4
 99              STA  LINE+2
100              LDA  TEMP
101              FSUB TEMPF
102              STA  TEMPF
103              ENTA 100
```

```
104                 STA  TEMP
105                 MUL  TEMP
106                 SLAX 5
107                 FLOT
108                 FMUL TEMPF
109                 FIX
110                 CHAR
111                 STX  LINE+4(2:5)
112                 OUT  LINE(TERM)
113                 JBUS *(TERM)
114                 ENT1 0
115                 ST1  LINE
116                 HLT
117 OVERFLOWST      ALF  OVERF
118                 ALF  LOW
119                 ORIG *+22
120 OVERFLOWEX      OUT  OVERFLOWST(TERM)
121                 JBUS *(TERM)
122                 HLT
123                 END  BEGIN
```

Ο τρόπος λειτουργίας του προγράμματος είναι αντίστοιχος με το προηγούμενο παράδειγμα. Η κύρια διαφορά είναι ότι σε αυτό το πρόγραμμα υπάρχουν και αριθμοί float, οπότε χρησιμοποιούνται αριθμητικές εντολές για αριθμούς float (π.χ. FADD αντι για ADD).



Σχήμα 4.2: Τα αποτελέσματα εκτέλεσης του 2ου παραδείγματος.