Προαιρετική Εργασία στο μάθημα
Σχεδίαση Γλωσσών και Μεταγλωττιστών

Χριστίνα Ισάκογλου - 2056
Σεπτέμβριος 2014

.:. ΕΡΓΑΛΕΙΑ ΚΑΙ ΑΡΧΕΙΑ .:.

Τα εργαλεία που χρησιμοποιήθηκαν για την κατασκευή του μεταγλωττιστή είναι τα : Bison, Flex και η συγγραφή του κώδικα έγινε στη γλώσσα C. Η εκτέλεση του τελικού κώδικα που παράγεται από το μεταγλωττιστή στη γλώσσα MIXAL έγινε στον προσομοιωτή της MIX, GNU MDK, σε λειτουργικό σύστημα GNU/Linux. Στα παρακάτω αρχεία βρίσκονται ο απαιτούμενος κώδικας.

- lex.l : Αναγνώριση λεξικών μονάδων της γλώσσας
- syd1.y : Κανόνες γραμματικής της γλώσσας.
- syd2.y : Σημασιολογικοί έλεγχοι (γίνεται include από το προηγούμενο αρχείο).
- syd3.c : Παραγωγή κώδικα σε MIXAL.
- defs.h : Δηλώσεις δομών δεδομένων και συναρτήσεων.
- Makefile : Αρχείο Makefile για μεταγλώττιση των αρχείων που παράγονται από τα εργαλεία flex και bison, καθώς και το MIXsyd3.
- makeMIXAL.sh : Shell script για την αυτοματοποίηση της όλης διαδικασίας μέχρι το τελικό βήμα της συνένωση των αρχείων που προκύπτουν από την παραγωγή κώδικα στο τελικό και ζητούμενο αρχείο .mixal.

Τα αρχεία που παράγονται κατά τη διαδικασία είναι τα εξής:

- lex.yy.c : Από τη λεξική ανάλυση.
- syd1.tab.c, syd1.tab.h : Από τη συντακτική ανάλυση.
- syd1.output : Επίσης από τη συντακτική ανάλυση για την περιγραφή του χώρου καταστάσεων.
- _ST : το εκτελέσιμο του μεταγλωττιστή.
- var.dat, code.cod : τα αρχεία παραγωγής κώδικα, τα οποία στη συνέχεια συνενώνονται (στο πρώτο δηλώνονται οι διευθύνσεις των μεταβλητών που θα χρησιμοποιηθούν με τη σειρά που αυτές δηλώνονται στο πρόγραμμα της γλώσσας – το δεύτερο περιέχει το κύριο μέρος του κώδικα του προγράμματος).

.:. ΠΕΡΙΓΡΑΦΗ .:.

Συντακτική Ανάλυση

- Σε περίπτωση ασάφειας για την αντιμετώπιση συγκρούσεων, ο τρόπος επίλυσης βασίζεται στον άμεσο ορισμό της προτεραιότητας και της προσεταιριστικότητας (%left,%right,%nonassoc) του εκάστοτε τερματικού. Με αυτόν τον τρόπο το συντακτικό δέντρο για τις αριθμητικές πράξεις

κατασκευάζεται έτσι όπως αναμένεται και η ασάφεια του μετέωρου else ακολουθεί τον κανόνα «Κάθε else συνοδεύει το πλησιέστερο από τα προηγούμενα ασυνόδευτα block then».

- Κανόνες που προστέθηκαν στη δοσμένη γραμματική αφορούν την εμφάνιση καμιάς, μίας φοράς ή παραπάνω μη τερματικών. Οι κανόνες αυτοί είναι πάντα αριστερά αναδρομικοί.

Παραγωγή κώδικα

- Η μηχανή MIX παρέχει 4000 διευθύνσεις μνήμης. Στα πλαίσια της παραγωγής κώδικα θεωρήθηκαν οι παρακάτω συμβάσεις:
    - Οι 500 πρώτες θέσεις μνήμης χρησιμοποιούνται αποκλειστικά για την αποθήκευση μεταβλητών.
    - Από την 500 μέχρι την 599 δημιουργείται μια προσομοίωση στοίβας, καθώς στη θέση 500(μόνο) αποθηκεύονται τιμές ως προς όφελος της μεταφοράς τους για μία μόνο φορά (πχ από έναν καταχωρητή σε έναν άλλο, καθώς εντολές μεταφοράς ανάμεσα σε καταχωρητές δεν υποστηρίζονται). Από τη θέση 501 μέχρι και το τέλος της προαναφερόμενης έκτασης αποθηκεύονται με παρόμοιο τρόπο όπως σε μια στοίβα τιμές οι οποίες προηγουμένως βρίσκονταν σε καταχωρητή αλλά αποτελούν αριστερό απόγονο κόμβου. Ο καταχωρητής r1 ρυθμίζεται με τέτοιον τρόπο έτσι ώστε να παρέχει κάθε στιγμή ένα offset ως προς την κορυφή της στοίβας(αναφορικά με τη διεύθυνση 500).
    - Από την 600 μέχρι και την 999 παρέχεται χώρος ως buffer για την εκτύπωση στον “Printer”.
    - Το κύριο πρόγραμμα ξεκινάει από τη διεύθυνση 1000.
    - Κάθε στιγμή ο καταχωρητής που θεωρείται ότι χρησιμοποιείται είναι ο rA. Ακόμα και σε περιπτώσεις που το αποτέλεσμα μιας πράξης αποθηκεύεται σε κάποιον άλλο, τότε αυτό στη συνέχεια μεταφέρεται στον rA.
- Η ροή του ελέγχου γίνεται πρωτίστως με τη χρήση ετικετών, που δηλώνουν τη διεύθυνση που πρέπει να μεταφερθεί ο έλεγχος, και την ύπαρξη μηδενικής(false) ή μη μηδενικής τιμής(true) στον καταχωρητή rA, γεγονός που καθορίζει τη μεταφορά ή μη μεταφορά στην αναφερόμενη διεύθυνση. Οι λογικές εκφράσεις είναι αυτές που οδηγούν στην ύπαρξη μηδενικής ή μη μηδενικής τιμής στον καταχωρητή rA(εκτός από τις περιπτώσεις που για συνθήκη τίθεται ένας τυχαίος αριθμός ή μεταβλητή), οι οποίες με τη σειρά τους βασίζονται στις σημαίες G,E,L που η μηχανή MIX σηκώνει ανάλογα με το αποτέλεσμα της σύγκρισης που έχει προηγηθεί.
- Στις επαναλήψεις είτε με while, είτε με for, σημαντικό ρόλο παίζουν και οι τοπικές μεταβλητές της MIX(1H-1B) για την υλοποίηση των break

statements, καθώς εμπεριέχοντας την έννοιας της τοπικότητας βοηθούν ούτως ώστε οι προαναφερόμενες εντολές να γίνονται ως προς το εσωτερικότερο block στο οποίο αυτές βρίσκονται εμφωλευμένες.

- Προσοχή χρειάζεται η υλοποίηση της εντολής continue. Ο παρακάτω κώδικας:

```
FOR1{
        FOR2{
        }
        CONTINUE;
        …
}
```

θα πρέπει να οδηγήσει τον έλεγχο στην FOR1, ξεκινώντας τον επόμενο κύκλο επανάληψης και όχι της FOR2 αυτό επιτυγχάνεται με χρήση μετρητή που δείχνει πάντα το id του for block μέσα στο οποίο βρισκόμαστε, ούτως ώστε ενδεχόμενο continue να μας μεταφέρει στην αρχή (πιο σωστά στον κώδικα που υλοποιεί το βήμα της επανάληψης) του τρέχοντος block.

.:. ΠΑΡΑΔΕΙΓΜΑΤΑ .:.

_____ Ανάθεση Τιμής _____

tAssignment

```
{
    var a,b,c,d:int;
    a = 7;
    print a;
```

0007

```
    b = 1;
    print b;
```

0001

```
    c = b;
    print c;
```

0001

```
    b += 2;
    print b;
```

0003

```
    c *= 7;
    print c;
```

0007

```
    c %= 2;
    print c;
```

0001

Στρογγυλοποίηση πάντα προς το μικρότερο ακέραιο

```
    a = 2;
    a *=(4-2);
    print a;
}
```

0004

```
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
b     EQU   1
c     EQU   2
d     EQU   3
      ORIG  1000-50
EXCT  ALF   "EXCEP"
      ALF   "TION "
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
      ORIG  1000
START ENT1  0
      LDA   =7=
      STA   a
      LDA   a
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)  Printing
      LDA   =1=
      STA   b
      LDA   b
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)  Printing
      ENT1  c
      MOVE  b(1)
      LDA   c
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)  Printing
      LDA   b
      ADD   =2=
      STA   b
      LDA   b
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)  Printing
      LDA   c
      MUL   =7=
```

```
        STX   BUFFER0
        LDA   BUFFER0
        STA   c
        LDA   c
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)  Printing
        LDA   =0=
        LDX   c
        DIV   =2=
        STX   c
        LDA   c
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)  Printing
        LDA   =2=
        STA   a
        LDA   a
        MUL   =2=
        STX   BUFFER0
        LDA   BUFFER0
        STA   a
        LDA   a
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)  Printing
* End of program execution.
        HLT
* End of assembler's compilation.
        END   START
```


tBinary

```
{
    var a,b,c,d : int;
    a = 3+3;
    b = a-4;
    c = a-(b+3);
    print c;

    c =2*a;
    print c;
```

0001

0012

0003

0001

0003

0003

0002

```
     c = a/b;
     print c;

     c = (a+3)%b;
     print c;

     a = (c*b)+1;
     print a;

     a = 1;
     d = -2;
     b = a*(5+d);
     print b;

     c = -2;
     d = -c;
     print d;
}
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
b     EQU   1
c     EQU   2
d     EQU   3
     ORIG  1000-50
EXCT ALF   "EXCEP"
     ALF   "TION "
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
     ORIG  1000
START ENT1  0
     LDA   =6=
     STA   a
     LDA   a
     SUB   =4=
     STA   b
     ENTA  3
     ADD   b
     STA   BUFFER0
     LDA   a
     SUB   BUFFER0
     STA   c
     LDA   c
```

```
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     ENTA  2
     MUL   a
     STX   BUFFER0
     LDA   BUFFER0
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   =0=
     LDX   a
     DIV   b
     JOV   EXC1
     JSJ   *+3
EXC1 OUT   EXCT(PRINTER)
     HLT
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     ENTA  3
     ADD   a
     INC1  1
     STA   BUFFER0,1
     STA   BUFFER0
     LDA   =0=
     LDX   BUFFER0,1
     DIV   b
     DEC1  1
     JOV   EXC2
     JSJ   *+3
EXC2 OUT   EXCT(PRINTER)
     HLT
     STX   BUFFER0
     LDA   BUFFER0
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
```

```
     OUT   BUFFER1(PRINTER)  Printing
     LDA   c
     MUL   b
     STX   BUFFER0
     LDA   BUFFER0
     INC1  1
     STA   BUFFER0,1
     STA   BUFFER0
     LDA   BUFFER0,1
     ADD   =1=
     DEC1  1
     STA   a
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   =1=
     STA   a
     ENNA  2
     LDA   =-2=
     STA   d
     ENTA  5
     ADD   d
     MUL   a
     STX   BUFFER0
     LDA   BUFFER0
     STA   b
     LDA   b
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     ENNA  2
     LDA   =-2=
     STA   c
     LDAN  c
     STA   d
     LDA   d
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
* End of program execution.
     HLT
* End of assembler's compilation.
     END   START
```

tLogical

```
{
    var a,b,c,d,e: int;
    a = 2;
    b = 4;
    c = 3;
    d = b<=a;
    e = c==3;
    print (d+e);

    a = 1;
    b = a && (2<3);
    print b;

    a = 0;
    b = a && (2<3);
    print b;

    a = 1;
    b = 2;
    c = b && (a<b);
    print c;

    a = 1;
    b = 2;
    d = 1;
    c = d && b && (a<b);
    print c;

    a = 1;
    b = 2;
    d = 0;
    c = d || (a && b);
    print c;

    a = 1;
    b = 2;
    d = 0;
    c = d || !(a && b);
    print c;

    a = 1;
    b = 2;
    d = 0;
    c = !(a && d) || d;
    print c;
```

0001

0001

Διακόπτει τον υπολογισμό όταν ο αριστερός τελεστέος δίνει 0 για το AND

0000

Οποιαδήποτε μη μηδενική τιμή λαμβάνεται ως true

0001

0001

0001

0000

Διακόπτει τον υπολογισμό όταν ο αριστερός τελεστέος δίνει 1 για το OR

0001

```
}
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
b     EQU   1
c     EQU   2
d     EQU   3
e     EQU   4
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
      ORIG  1000
START ENT1  0
      LDA   =2=
      STA   a
      LDA   =4=
      STA   b
      LDA   =3=
      STA   c
      LDA   b
      CMPA  a
      JL    TRUE1
      JE    TRUE1
      LDA   =0=
      JMP   ENDCMP1
TRUE1 LDA   =1=
ENDCMP1    NOP
      STA   d
      LDA   c
      CMPA  =3=
      JE    TRUE2
      LDA   =0=
      JMP   ENDCMP2
TRUE2 LDA   =1=
ENDCMP2    NOP
      STA   e
      LDA   d
      ADD   e
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)
      LDA   =1=
      STA   a
      LDA   =2=
```

```
     CMPA  =3=
     JL    TRUE3
     LDA   =0=
     JMP   ENDCMP3
TRUE3LDA   =1=
ENDCMP3    NOP
     STA   BUFFER0
     LDA   a
     JAZ   FAND1
     LDA   BUFFER0
     JAZ   FAND1
     LDA   =1=
     JMP   EAND1
FAND1LDA   =0=
EAND1NOP
     STA   b
     LDA   b
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =0=
     STA   a
     LDA   =2=
     CMPA  =3=
     JL    TRUE4
     LDA   =0=
     JMP   ENDCMP4
TRUE4LDA   =1=
ENDCMP4    NOP
     STA   BUFFER0
     LDA   a
     JAZ   FAND2
     LDA   BUFFER0
     JAZ   FAND2
     LDA   =1=
     JMP   EAND2
FAND2LDA   =0=
EAND2NOP
     STA   b
     LDA   b
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
```

```
     LDA   =2=
     STA   b
     LDA   a
     CMPA  b
     JL    TRUE5
     LDA   =0=
     JMP   ENDCMP5
TRUE5LDA   =1=
ENDCMP5    NOP
     STA   BUFFER0
     LDA   b
     JAZ   FAND3
     LDA   BUFFER0
     JAZ   FAND3
     LDA   =1=
     JMP   EAND3
FAND3LDA   =0=
EAND3NOP
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
     LDA   =2=
     STA   b
     LDA   =1=
     STA   d
     LDA   d
     JAZ   FAND4
     LDA   b
     JAZ   FAND4
     INC1  1
     STA   BUFFER0,1
     LDA   =1=
     JMP   EAND4
FAND4LDA   =0=
EAND4NOP
     LDA   a
     CMPA  b
     JL    TRUE6
     LDA   =0=
     JMP   ENDCMP6
TRUE6LDA   =1=
ENDCMP6    NOP
```

```
     STA   BUFFER0
     LDA   BUFFER0,1
     JAZ   FAND5
     LDA   BUFFER0
     JAZ   FAND5
     DEC1  1
     LDA   =1=
     JMP   EAND5
FAND5LDA   =0=
EAND5NOP
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
     LDA   =2=
     STA   b
     LDA   =0=
     STA   d
     LDA   a
     JAZ   FAND6
     LDA   b
     JAZ   FAND6
     LDA   =1=
     JMP   EAND6
FAND6LDA   =0=
EAND6NOP
     STA   BUFFER0
     LDA   d
     JANZ  TOR1
     LDA   BUFFER0
     JANZ  TOR1
     LDA   =0=
     JMP   EOR1
TOR1 LDA   =1=
EOR1 NOP
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
```

```
     LDA   =2=
     STA   b
     LDA   =0=
     STA   d
     LDA   a
     JAZ   FAND7
     LDA   b
     JAZ   FAND7
     LDA   =1=
     JMP   EAND7
FAND7LDA   =0=
EAND7NOP
     JAZ   FNOT1
     LDA   =0=
     JMP   ENOT1
FNOT1LDA   =1=
ENOT1NOP
     STA   BUFFER0
     LDA   d
     JANZ  TOR2
     LDA   BUFFER0
     JANZ  TOR2
     LDA   =0=
     JMP   EOR2
TOR2 LDA   =1=
EOR2 NOP
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
     LDA   =2=
     STA   b
     LDA   =0=
     STA   d
     LDA   a
     JAZ   FAND8
     LDA   d
     JAZ   FAND8
     LDA   =1=
     JMP   EAND8
FAND8LDA   =0=
EAND8NOP
     JAZ   FNOT2
```

```
     LDA   =0=
     JMP   ENOT2
FNOT2LDA   =1=
ENOT2NOP
     INC1  1
     STA   BUFFER0,1
     LDA   BUFFER0,1
     JANZ  TOR3
     LDA   d
     JANZ  TOR3
     DEC1  1
     LDA   =0=
     JMP   EOR3
TOR3 LDA   =1=
EOR3 NOP
     STA   c
     LDA   c
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
* End of program execution.
     HLT
* End of assembler's compilation.
     END   START
```

Εντολές Συνθήκης

tIf

```
{
     var a,b:int;
     if (1) if (1) a=1;
     print a;

     a = 2;
     b = 1;
     if (a&&b) a=0;
     else a=1;
     print a;

     a = 2;
     b = 1;
     if (!a&&!b) a=0;
     else a=1;
     print a;

     a = 0;
```

0001

0000

0001

0010

0001
0002
0003

```
     b = 1;
     if ((a&&b)||b) a=10;
     print a;

     a = 1;
     if (a<=1){
          print 1;
          print 2;
     }
     else {
          print 0;
     }
     print 3;
}
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
b     EQU   1
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
     ORIG  1000
STARTENT1  0
     ENTA  1
     JAZ   EIF1
     ENTA  1
     JAZ   EIF2
     LDA   =1=
     STA   a
EIF2 NOP
EIF1 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =2=
     STA   a
     LDA   =1=
     STA   b
     LDA   a
     JAZ   FAND1
     LDA   b
     JAZ   FAND1
     LDA   =1=
```

```
     JMP   EAND1
FAND1LDA   =0=
EAND1NOP
     JAZ   EEIF3
     LDA   =0=
     STA   a
     JMP   EIF3
EEIF3NOP
     LDA   =1=
     STA   a
EIF3 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =2=
     STA   a
     LDA   =1=
     STA   b
     LDA   a
     JAZ   FNOT1
     LDA   =0=
     JMP   ENOT1
FNOT1LDA   =1=
ENOT1NOP
     INC1  1
     STA   BUFFER0,1
     LDA   b
     JAZ   FNOT2
     LDA   =0=
     JMP   ENOT2
FNOT2LDA   =1=
ENOT2NOP
     STA   BUFFER0
     LDA   BUFFER0,1
     JAZ   FAND2
     LDA   BUFFER0
     JAZ   FAND2
     DEC1  1
     LDA   =1=
     JMP   EAND2
FAND2LDA   =0=
EAND2NOP
     JAZ   EEIF4
     LDA   =0=
     STA   a
```

```
     JMP   EIF4
EEIF4NOP
     LDA   =1=
     STA   a
EIF4 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =0=
     STA   a
     LDA   =1=
     STA   b
     LDA   a
     JAZ   FAND3
     LDA   b
     JAZ   FAND3
     INC1  1
     STA   BUFFER0,1
     LDA   =1=
     JMP   EAND3
FAND3LDA   =0=
EAND3NOP
     LDA   BUFFER0,1
     JANZ  TOR1
     LDA   b
     JANZ  TOR1
     DEC1  1
     LDA   =0=
     JMP   EOR1
TOR1 LDA   =1=
EOR1 NOP
     JAZ   EIF5
     LDA   =10=
     STA   a
EIF5 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)
     LDA   =1=
     STA   a
     LDA   a
     CMPA  =1=
     JL    TRUE1
```

```
        JE    TRUE1
        LDA   =0=
        JMP   ENDCMP1
TRUE1   LDA   =1=
ENDCMP1 NOP
        JAZ   EEIF6
        ENTA  1
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)
        ENTA  2
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)
        JMP   EIF6
EEIF6   NOP
        ENTA  0
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)
EIF6    NOP
        ENTA  3
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)
* End of program execution.
        HLT
* End of assembler's compilation.
        END   START
```

_____Εντολές Επανάληψης _____

tRepeat

```
{
    var a,i:int;
    a = 1;
    while (a<=2){
        print a;
        a = a+1;
    }

    a = 1;
    while (a<=3){
        print a;
```

0001
0002

0001
0002

```
            a = a+1;
            if (a == 3) break;
      }

      for (a=1;a<=3;a+=1){
            if (a==1) continue;
            print a;
      }

      for (a=1;a<=3;a+=1){
            if (a==2) break;
            print a;
      }

      i = 0;
      for(;i<=1;){
            print i;
            i+=1;
      }

}
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
i     EQU   1
      ORIG  1000-50
EXCT  ALF   "EXCEP"
      ALF   "TION "
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
      ORIG  1000
START ENT1  0
      LDA   =1=
      STA   a
1H    NOP
SWH1  NOP   EWH1
      LDA   a
      CMPA  =2=
      JL    TRUE1
      JE    TRUE1
      LDA   =0=
      JMP   ENDCMP1
TRUE1 LDA   =1=
ENDCMP1    NOP
```

0002
0003

0001

0000
0001

```
     JAZ   EWH1
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     ENTA  1
     ADD   a
     STA   a
     JMP   SWH1
EWH1 NOP
1H   NOP
     LDA   =1=
     STA   a
1H   NOP
SWH2 NOP   EWH2
     LDA   a
     CMPA  =3=
     JL    TRUE2
     JE    TRUE2
     LDA   =0=
     JMP   ENDCMP2
TRUE2LDA   =1=
ENDCMP2    NOP
     JAZ   EWH2
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     ENTA  1
     ADD   a
     STA   a
     LDA   a
     CMPA  =3=
     JE    TRUE3
     LDA   =0=
     JMP   ENDCMP3
TRUE3LDA   =1=
ENDCMP3    NOP
     JAZ   EIF1
     JMP   1F
EIF1 NOP
     JMP   SWH2
EWH2 NOP
1H   NOP
IFOR1NOP
```

```
     LDA   =1=
     STA   a
SFOR1NOP
     LDA   a
     CMPA  =3=
     JL    TRUE4
     JE    TRUE4
     LDA   =0=
     JMP   ENDCMP4
TRUE4LDA   =1=
ENDCMP4    NOP
     JAZ   EFOR1
     LDA   a
     CMPA  =1=
     JE    TRUE5
     LDA   =0=
     JMP   ENDCMP5
TRUE5LDA   =1=
ENDCMP5    NOP
     JAZ   EIF2
     JMP   SPFOR1
EIF2 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
SPFOR1     NOP
     LDA   a
     ADD   =1=
     STA   a
     JSJ   SFOR1
EFOR1NOP
1H   NOP
IFOR2NOP
     LDA   =1=
     STA   a
SFOR2NOP
     LDA   a
     CMPA  =3=
     JL    TRUE6
     JE    TRUE6
     LDA   =0=
     JMP   ENDCMP6
TRUE6LDA   =1=
ENDCMP6    NOP
     JAZ   EFOR2
```

```
     LDA   a
     CMPA  =2=
     JE    TRUE7
     LDA   =0=
     JMP   ENDCMP7
TRUE7LDA   =1=
ENDCMP7    NOP
     JAZ   EIF3
     JMP   1F
EIF3 NOP
     LDA   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
SPFOR2     NOP
     LDA   a
     ADD   =1=
     STA   a
     JSJ   SFOR2
EFOR2NOP
1H   NOP
     LDA   =0=
     STA   i
IFOR3NOP
     NOP
SFOR3NOP
     LDA   i
     CMPA  =1=
     JL    TRUE8
     JE    TRUE8
     LDA   =0=
     JMP   ENDCMP8
TRUE8LDA   =1=
ENDCMP8    NOP
     JAZ   EFOR3
     LDA   i
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   i
     ADD   =1=
     STA   i
SPFOR3     NOP
     NOP
     JSJ   SFOR3
```

```
EFOR3NOP
1H   NOP
* End of program execution.
     HLT
* End of assembler's compilation.
     END   START
```

tRepeatNestedWhile

```
{
     var a,b,i,j :int;
     a=0;
     while (a!=1){
           print (a+1);
           b=0;
           while (b<=1){
                 b+=1;
                 print (b);
                 if (b==1) a=1;
           }
     }

     a=0;
     while (a!=1){
           print (a+1);
           b=0;
           while (b<=1){
                 b+=1;
                 print (b);
                 if (b==1) break;
           }
           a+=1;
     }

}
| Παραγωγή κώδικα |
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a    EQU   0
b    EQU   1
i    EQU   2
j    EQU   3
     ORIG  1000-50
EXCT ALF   "EXCEP"
     ALF   "TION "
* Start of program execution.
```

```
0001
0001
0002
```

```
0001
0001
```

```
* [LOC]    OP    [OPERAND]   [comment]
     ORIG  1000
START ENT1  0
     LDA   =0=
     STA   a
1H   NOP
SWH1 NOP   EWH1
     LDA   a
     CMPA  =1=
     JNE   TRUE1
     LDA   =0=
     JMP   ENDCMP1
TRUE1 LDA   =1=
ENDCMP1    NOP
     JAZ   EWH1
     ENTA  1
     ADD   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   =0=
     STA   b
1H   NOP
SWH2 NOP   EWH2
     LDA   b
     CMPA  =1=
     JL    TRUE2
     JE    TRUE2
     LDA   =0=
     JMP   ENDCMP2
TRUE2 LDA   =1=
ENDCMP2    NOP
     JAZ   EWH2
     LDA   b
     ADD   =1=
     STA   b
     LDA   b
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   b
     CMPA  =1=
     JE    TRUE3
     LDA   =0=
     JMP   ENDCMP3
```

```
TRUE3LDA   =1=
ENDCMP3    NOP
     JAZ   EIF1
     LDA   =1=
     STA   a
EIF1 NOP
     JMP   SWH2
EWH2 NOP
1H   NOP
     JMP   SWH1
EWH1 NOP
1H   NOP
     LDA   =0=
     STA   a
1H   NOP
SWH3 NOP   EWH3
     LDA   a
     CMPA  =1=
     JNE   TRUE4
     LDA   =0=
     JMP   ENDCMP4
TRUE4LDA   =1=
ENDCMP4    NOP
     JAZ   EWH3
     ENTA  1
     ADD   a
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   =0=
     STA   b
1H   NOP
SWH4 NOP   EWH4
     LDA   b
     CMPA  =1=
     JL    TRUE5
     JE    TRUE5
     LDA   =0=
     JMP   ENDCMP5
TRUE5LDA   =1=
ENDCMP5    NOP
     JAZ   EWH4
     LDA   b
     ADD   =1=
     STA   b
     LDA   b
```

```
        ENTX  0
        CHAR
        STX   BUFFER1(1:4)
        OUT   BUFFER1(PRINTER)  Printing
        LDA   b
        CMPA  =1=
        JE    TRUE6
        LDA   =0=
        JMP   ENDCMP6
TRUE6   LDA   =1=
ENDCMP6 NOP
        JAZ   EIF2
        JMP   1F
EIF2    NOP
        JMP   SWH4
EWH4    NOP
1H      NOP
        LDA   a
        ADD   =1=
        STA   a
        JMP   SWH3
EWH3    NOP
1H      NOP
* End of program execution.
        HLT
* End of assembler's compilation.
        END   START
```

tRepeatNestedFor

```
{
      var i,j: int;
      for(i=1;i<=3;i+=1){
            for(j=1;j<=3;j+=1){
                  if (j==2) break;
                  print i;
            }
      }
```

0001
0002
0003

```
      for(i=1;i<=3;i+=1){
            for(j=1;j<=3;j+=1){
                  if (j==2) continue;
                  print i;
            }
      }

}
```

0001
0001
0002
0002
0003
0003

```
| Παραγωγή κώδικα |
* Declaration of space for each variable.
```

```
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
i    EQU   0
j    EQU   1
     ORIG  1000-50
EXCT ALF   "EXCEP"
     ALF   "TION "
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
     ORIG  1000
STARTENT1  0
IFOR1NOP
     LDA   =1=
     STA   i
SFOR1NOP
     LDA   i
     CMPA  =3=
     JL    TRUE1
     JE    TRUE1
     LDA   =0=
     JMP   ENDCMP1
TRUE1LDA   =1=
ENDCMP1    NOP
     JAZ   EFOR1
IFOR2NOP
     LDA   =1=
     STA   j
SFOR2NOP
     LDA   j
     CMPA  =3=
     JL    TRUE2
     JE    TRUE2
     LDA   =0=
     JMP   ENDCMP2
TRUE2LDA   =1=
ENDCMP2    NOP
     JAZ   EFOR2
     LDA   j
     CMPA  =2=
     JE    TRUE3
     LDA   =0=
     JMP   ENDCMP3
TRUE3LDA   =1=
ENDCMP3    NOP
     JAZ   EIF1
     JMP   1F
```

```
EIF1 NOP
     LDA   i
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
SPFOR2     NOP
     LDA   j
     ADD   =1=
     STA   j
     JSJ   SFOR2
EFOR2NOP
1H   NOP
SPFOR1     NOP
     LDA   i
     ADD   =1=
     STA   i
     JSJ   SFOR1
EFOR1NOP
1H   NOP
IFOR3NOP
     LDA   =1=
     STA   i
SFOR3NOP
     LDA   i
     CMPA  =3=
     JL    TRUE4
     JE    TRUE4
     LDA   =0=
     JMP   ENDCMP4
TRUE4LDA   =1=
ENDCMP4    NOP
     JAZ   EFOR3
IFOR4NOP
     LDA   =1=
     STA   j
SFOR4NOP
     LDA   j
     CMPA  =3=
     JL    TRUE5
     JE    TRUE5
     LDA   =0=
     JMP   ENDCMP5
TRUE5LDA   =1=
ENDCMP5    NOP
     JAZ   EFOR4
     LDA   j
```

```
     CMPA  =2=
     JE    TRUE6
     LDA   =0=
     JMP   ENDCMP6
TRUE6LDA   =1=
ENDCMP6    NOP
     JAZ   EIF2
     JMP   SPFOR4
EIF2 NOP
     LDA   i
     ENTX  0
     CHAR
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
SPFOR4     NOP
     LDA   j
     ADD   =1=
     STA   j
     JSJ   SFOR4
EFOR4NOP
1H   NOP
SPFOR3     NOP
     LDA   i
     ADD   =1=
     STA   i
     JSJ   SFOR3
EFOR3NOP
1H   NOP
* End of program execution.
     HLT
* End of assembler's compilation.
     END   START
```

tRepeatNestedWhileFor

```
{
     var a,b :int;
     a=0;
     while (a!=1){
           print (a+1);
           for (b=0;b<=2;b+=1){
                 print (b);
                 if (b==1) break;
           }
           a+=1;
     }
}
```

```
0001
0000
0001
```

| Παραγωγή κώδικα |

```
* Declaration of space for each variable.
BUFFER0    EQU   500
BUFFER1    EQU   600
PRINTER    EQU   18
a     EQU   0
b     EQU   1
      ORIG  1000-50
EXCT  ALF   "EXCEP"
      ALF   "TION "
* Start of program execution.
* [LOC]    OP    [OPERAND]   [comment]
      ORIG  1000
START ENT1  0
      LDA   =0=
      STA   a
1H    NOP
SWH1  NOP   EWH1
      LDA   a
      CMPA  =1=
      JNE   TRUE1
      LDA   =0=
      JMP   ENDCMP1
TRUE1 LDA   =1=
ENDCMP1    NOP
      JAZ   EWH1
      ENTA  1
      ADD   a
      ENTX  0
      CHAR
      STX   BUFFER1(1:4)
      OUT   BUFFER1(PRINTER)  Printing
IFOR1 NOP
      LDA   =0=
      STA   b
SFOR1 NOP
      LDA   b
      CMPA  =2=
      JL    TRUE2
      JE    TRUE2
      LDA   =0=
      JMP   ENDCMP2
TRUE2 LDA   =1=
ENDCMP2    NOP
      JAZ   EFOR1
      LDA   b
      ENTX  0
      CHAR
```

```
     STX   BUFFER1(1:4)
     OUT   BUFFER1(PRINTER)  Printing
     LDA   b
     CMPA  =1=
     JE    TRUE3
     LDA   =0=
     JMP   ENDCMP3
TRUE3LDA   =1=
ENDCMP3    NOP
     JAZ   EIF1
     JMP   1F
EIF1 NOP
SPFOR1     NOP
     LDA   b
     ADD   =1=
     STA   b
     JSJ   SFOR1
EFOR1NOP
1H   NOP
     LDA   a
     ADD   =1=
     STA   a
     JMP   SWH1
EWH1 NOP
1H   NOP
* End of program execution.
     HLT
* End of assembler's compilation.
     END   START
```