Ενδιάμεσος Κώδικας

Εντολές:

Assign:	:=, x, _, z	z:=x
Unconditional jump:	jump, _, _, z	jump στην εντολή z
Conditional jump:	relop, x, y, z	relop: <, >, <>, <=, >= και αν x relop y κάνε jump στην εντολή k αλλιώς πάνε παρακάτω
Σημείωση μεταβλητών για συναρτήσεις:	par, x, m, _	x: μεταβλητή που περνάει σαν παράμετρος m: τρόπος περάσματος CV: με τιμή REF: με αναφορά RET: επιστροφή
Κλήση συνάρτησης:	call, name, _, _	
Αρχή προγράμματος ή υποπρογράμματος με όνομα name:	begin_block, name, _, _	
Τέλος προγράμματος ή υποπρογράμματος με όνομα name:	end_block, name, _, _	
Τέλος προγράμματος:	halt, _, _, _	

Οι εντολές είναι μία συνδεδεμένη λίστα από 4άδες

```
typedef struct linkedquad{
    char *label[40];
    char *op[40];
    char *op1[40];
    char *op2[40];
    char *op3[40];
    struct linkedquad *next;
}quad;

Global var: quad *aquad=NULL;
Global var: quad *lastquad=NULL;
```

- `genquad(char *a, char *b, char *c, char *d)`

  Δημιουργεί μία τετράδα και την ενώνει στο τέλος της συνδεδεμένης λίστας
  Πχ genquad("*","3","5","c") ->

102	*	3	5	c

  Το label κάθε quad θα το βρίσκει η nextquad()

- `nextquad()`

  Επιστρέφει ως string τον αριθμό της επόμενης τετράδας που θα παραχθεί.
  Μετατρέπει δηλαδή το label του lastquad σε ακέραιο, τον αυξάνει κατά 1 και τον επιστρέφει ως string.

- `newTemp()`

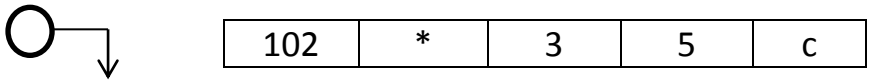
  Επιστρέφει την επόμενη προσωρινή μεταβλητή σε string με χρήση ενός global counter.
  Αποθήκευση του string σε global var

```
Global var: int temp_cnt=0;
Global var: char *temp_var[40];
```

  Οι μεταβλητές θα είναι της μορφής T_1, T_2, T_3…

- `emptylist()`

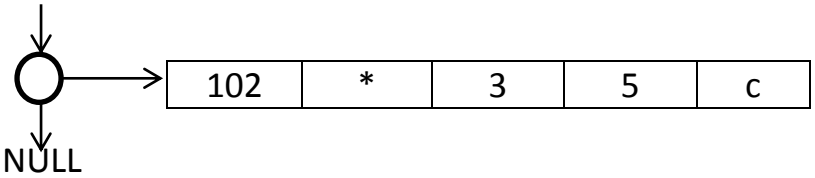
  Δημιουργεί μία κενή λίστα τετράδων.



  Δηλαδή θα έχουμε έναν pointer quad *quad_pbt=NULL, ο οποίος δε θα δείχνει πουθενά, αλλά προφανώς θα μπορεί να δείξει όταν το θέλουμε σε κάποιο quad aquad.

- `makelist(char *label)`

  Αναζητά την τετράδα με το label της παραμέτρου. Στη συνέχεια ενώνει τον κόμβο με την τετράδα και βάζει τον κόμβο σε μία λίστα δεικτών με μόνο κόμβο αυτόν.




```
typedef struct listofquads{
    quad *link;
    struct listofquads *next;
}quadlist;

Global var: quadlist *aquadlist1=NULL;
Global var: quadlist *aquadlist2=NULL;
```

- `merge(quadlist list1, quadlist list2)`

  Παίρνει τη λίστα list2 και την κολλάει στο τέλος της list1. Πλέον η list1 είναι η λίστα που περιέχει και τις δύο λίστες.

- `backpatch(quadlist list, char *x)`

  Διατρέχει τη λίστα list και κάνει το op3 ίσο με x, δηλαδή:
  `list.quad.op[3]=x` (προφανώς θα χρειαστούμε και την strcpy)

  Στο τέλος της διαδικασίας η λίστα γίνεται free.