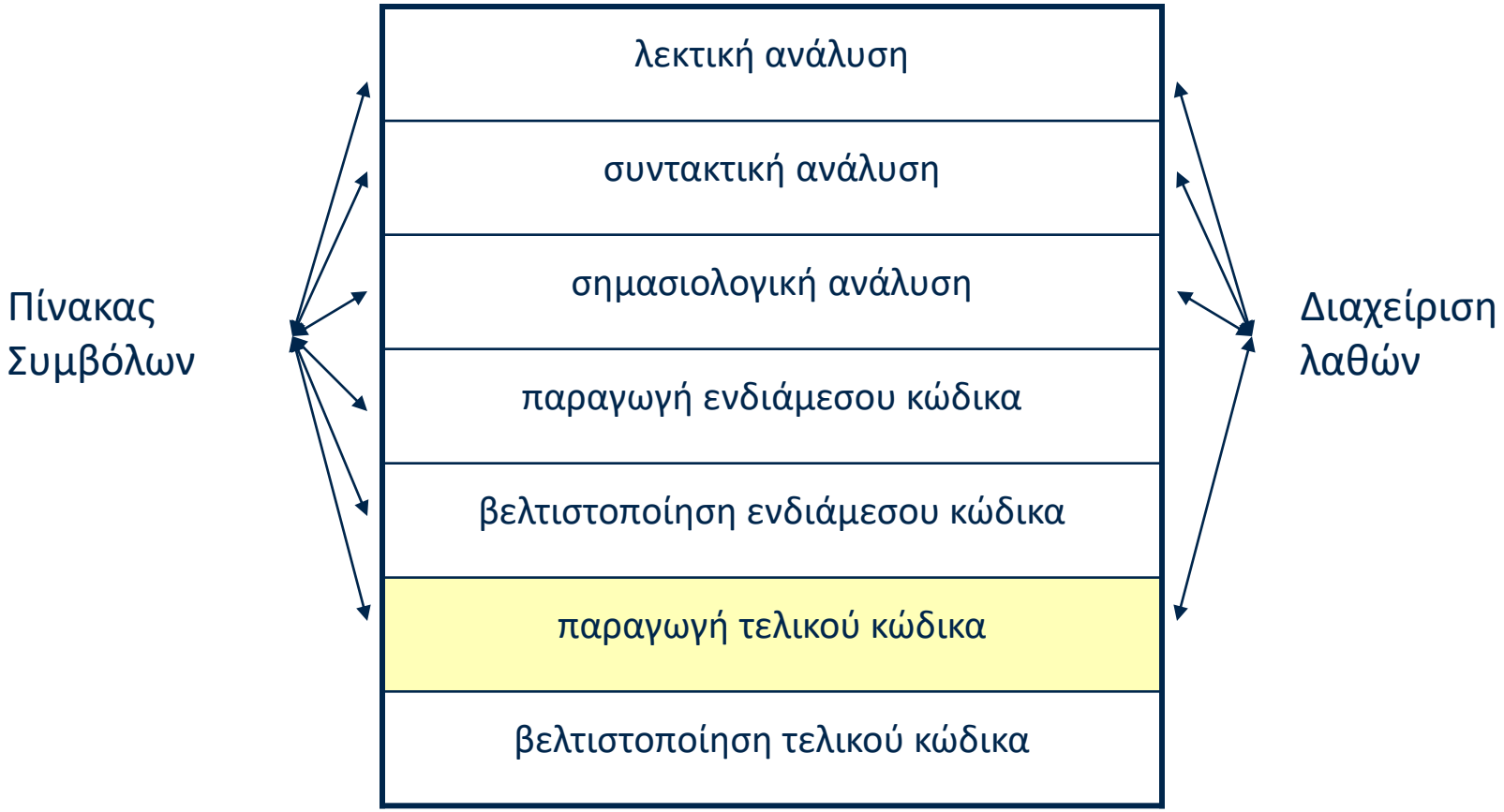
Παραγωγή Τελικού Κώδικα για την Αρχιτεκτονική RISC-V

Διαλέξεις στο μάθημα: Μεταφραστές
Γεώργιος Μανής

Οι Φάσεις της Μεταγλώττισης



Παραγωγή Τελικού Κώδικα

Παραγωγή Τελικού Κώδικα

- Από **κάθε εντολή ενδιάμεσου** κώδικα παράγουμε τις **αντίστοιχες εντολές του τελικού** κώδικα
- Κύριες ενέργειες στη φάση αυτή:
  - οι **μεταβλητές απεικονίζονται στην μνήμη** (στοίβα)
  - το **πέρασμα παραμέτρων** και η **κλήση συναρτήσεων**
- θα δημιουργήσουμε κώδικα για τον επεξεργαστή RISC-V

Η Αρχιτεκτονική RISC-V - Καταχωρητές

- Καταχωρητές που θα μας φανούν χρήσιμοι:
  - καταχωρητής που έχει πάντα τιμή **0**: zero
  - καταχωρητές **προσωρινών τιμών**: t0…t6
  - καταχωρητές οι τιμές των οποίων διατηρούνται **ανάμεσα σε κλήσεις** συναρτήσεων: s1…s11
  - καταχωρητές **ορισμάτων**: a0…a7
  - **stack pointer** sp
  - **frame pointer** fp
  - **return address** ra
  - **global pointer** gp

Απευθείας εκχώρηση τιμής

- Η απευθείας εκχώρηση αριθμητικής σταθεράς σε έναν καταχωρητή γίνεται με την ψευδοεντολή `li`:

```
li reg,int # reg = int
# reg: destination register
# int: arithmetic integer constant
```

Για παράδειγμα η εντολή

```
li t0,3
```

εκχωρεί το 3 στον καταχωρητή `t0`.

Μεταφορά τιμής

- Η μεταφορά τιμής από έναν καταχωρητή σε έναν άλλο γίνεται με την ψευδοεντολή `mv`:

```
mv reg1,reg2 # reg1 = reg2
# reg1 : destination register
# reg2 : source register
```

Στο παρακάτω πρόγραμμα οι καταχωρητές `t1` και `t2` εναλλάσσουν τις τιμές τους, μέσω του καταχωρητή `t0`:

```
mv t0,t1
mv t1,t2
mv t2,t0
```

Αριθμητικές πράξεις

- Για τις τέσσερις αριθμητικές πράξεις ανάμεσα σε καταχωρητές είναι διαθέσιμες οι εξής εντολές:

```
add reg, reg1, reg2     # reg = reg1 + reg2
sub reg, reg1, reg2     # reg = reg1 - reg2
mul reg, reg1, reg2     # reg = reg1 * reg2
div reg, reg1, reg2     # reg = reg1 / reg2
    # reg: destination register
    # reg1,reg2: registers (operands)
```

Αριθμητικές πράξεις

- Πρόσθεση μίας ακέραιας σταθεράς σε έναν καταχωρητή γίνεται με την addi

```
addi target_reg, source_reg, int # target_reg =
source_reg + int
# target_reg , source_reg : registers
# int: arithmetic integer constant
```

Αριθμητικές πράξεις

- Η διαίρεση μεταξύ ακεραίων επιστρέφει ακέραιο αριθμό, ενώ υπάρχει και η `rem` η οποία επιστρέφει το υπόλοιπο της διαίρεσης.

- Στο παρακάτω πρόγραμμα οι καταχωρητές `t1` και `t2` εναλλάσσουν τις τιμές τους, χωρίς τη χρήση του καταχωρητή `t0`:

```
add t1, t1, t2
sub t2, t1, t2
sub t1, t1, t2
```

Πρόσβαση στη μνήμη

- Με τις lw και sw
- Μας ενδιαφέρει τη πρόσβαση μέσω καταχωρητή

```
lw reg1,offset(reg2)    # reg1 = [reg2 + offset]
    # reg1: destination register
    # reg2: base register
    # offset: distance from reg2
sw reg1,offset(reg2)    # [reg2 + offset] = reg1
    # reg1: source register
    # reg2: base register
    # offset: distance from reg2
```

Πρόσβαση στη μνήμη

- Ένας άλλος τρόπος πρόσβασης είναι η πρόσβαση μέσω καταχωρητή, χωρίς να δηλωθεί κάποιο `offset`:

```
lw reg1,(reg2)    # reg1 = [reg2]
    # reg1: destination register
    # reg2: base register
    # offset: distance from reg2
sw reg1,(reg2)    # [reg2] = reg1
    # reg1: source register
    # reg2: base register
    # offset: distance from reg2
```

- Ο συμβολισμός `(t0)` είναι ισοδύναμος με τον συμβολισμό `0(t0)`

Άλματα

- Εντολές που θα μας φανούν χρήσιμες για άλματα:

b label	**branch** to label
beq t1,t2,label	jump to label **if t1=t2**
blt t1,t2,label	jump to label **if t1<t2**
bgt t1,t2,label	jump to label **if t1>t2**
ble t1,t2,label	jump to label **if t1<=t2**
bge t1,t2,label	jump to label **if t1>=t2**
bne t1,t2,label	jump to label **if t1<>t2**

Κλήσεις συναρτήσεων

- Εντολές που θα μας φανούν χρήσιμες στην κλήση συναρτήσεων:

j label	**jump** to label
jal label	**κλήση** συνάρτησης
jr ra	**άλμα** στη διεύθυνση που έχει ο **καταχωρητής**, στο παράδειγμα είναι ο ra που έχει την διεύθυνση επιστροφής συνάρτησης

Είσοδος και έξοδος δεδομένων

- Είσοδος δεδομένων

```
li a7,5
ecall
```

Θα διαβαστεί από το πληκτρολόγιο ακέραιος αριθμός και θα τοποθετηθεί στον a0

- Έξοδος δεδομένων

```
li a0,44
li a7,1
ecall
```

Το 44 εμφανίζεται στην οθόνη

Είσοδος και έξοδος δεδομένων

- Ο παρακάτω κώδικας τυπώνει μία αλλαγή γραμμής

```
    .data
str_nl: .asciz "\n"
    .text
la a0,str_nl
li a7,4
ecall
```

Τερματισμός προγράμματος

- Τερματισμός προγράμματος με επιστροφή τιμής 0

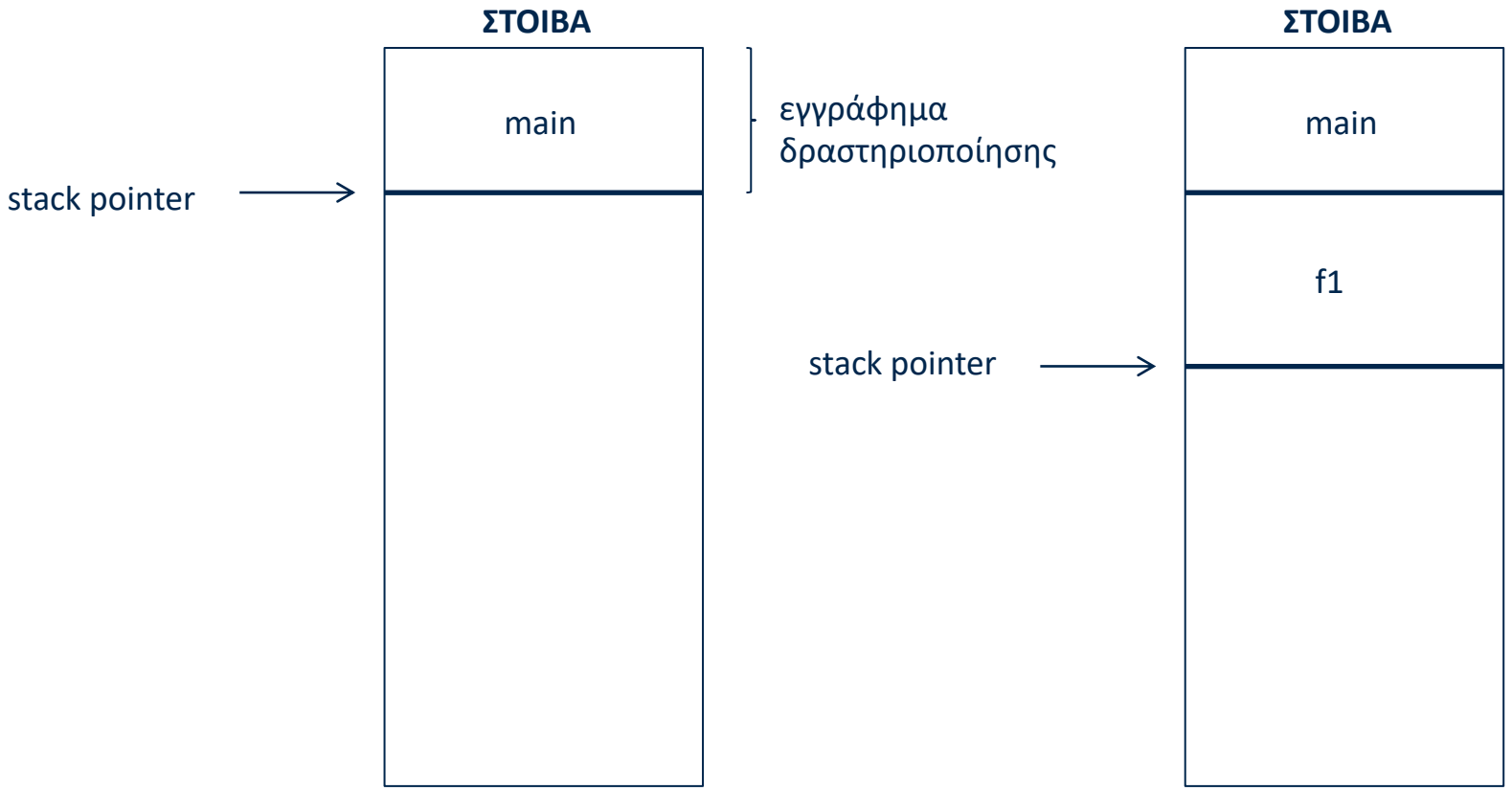
```
li a0,0
li a7,93
ecall
```

Τι κάνει?

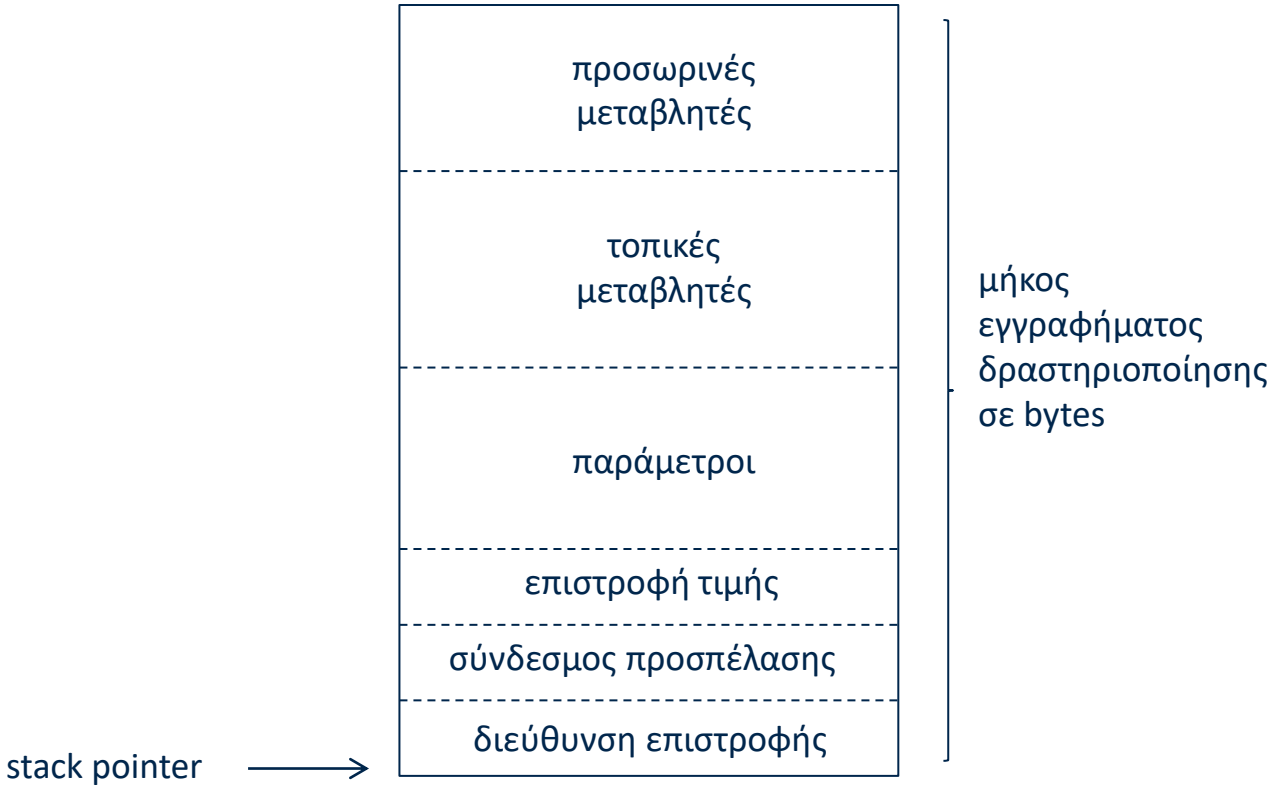
```
main:

input:
    li a7,5
    ecall

do_it:
    mv t0,a0
    addi t0,t0,t0
```

```
print:
        mv a0,t0
        li a7,1
        ecall
        la a0,str_nl
        li a7,4
        ecall
exit:
        li a0,0
        li a7,93
        ecall
```

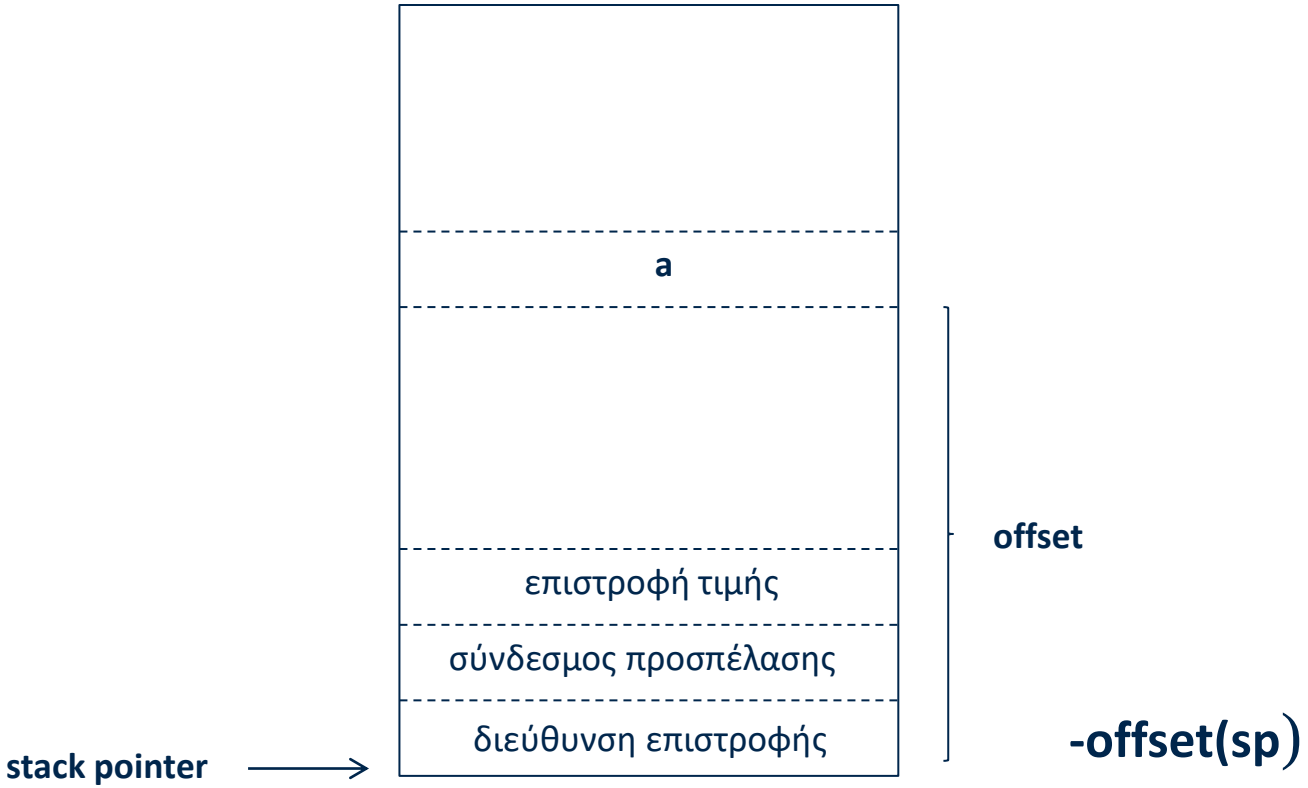
Εκτέλεση Προγράμματος



Εγγράφημα Δραστηριοποίησης



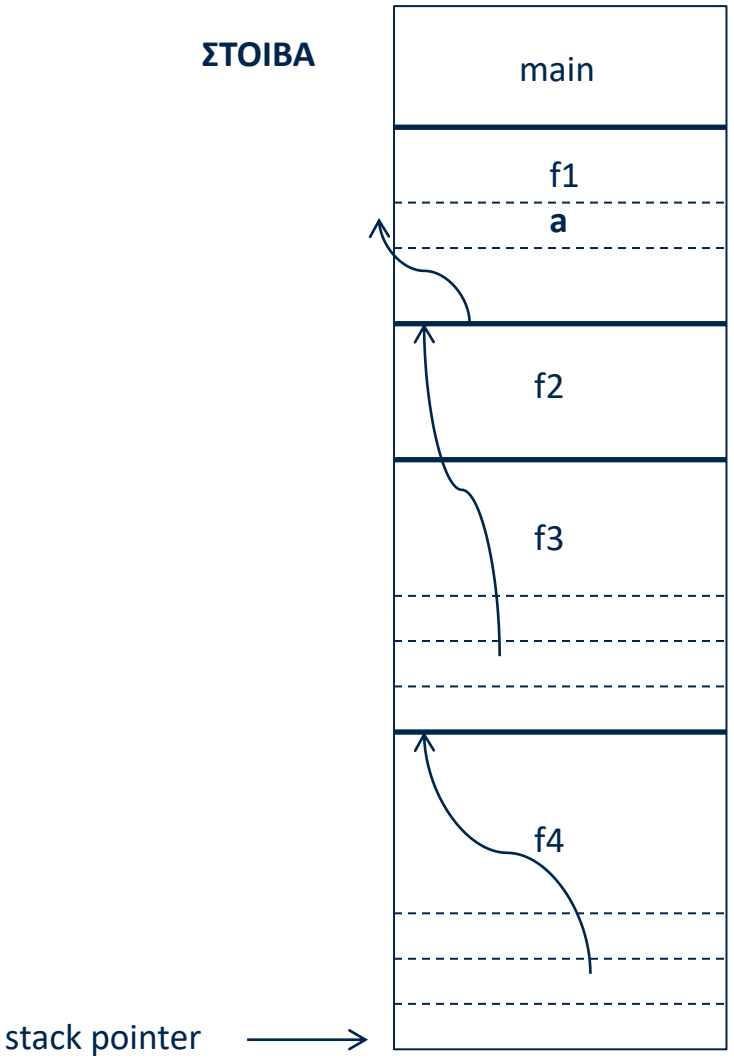
Θέση Μεταβλητής με βάση το Offset



Βοηθητικές Συναρτήσεις - gnvlcode

- μεταφέρει στον t0 την διεύθυνση μιας **μη τοπικής** μεταβλητής
- από τον πίνακα συμβόλων βρίσκει πόσα επίπεδα επάνω βρίσκεται η μη τοπική μεταβλητή και μέσα από τον σύνδεσμο προσπέλασης την εντοπίζει

Βοηθητικές Συναρτήσεις - gnvlcode



Βοηθητικές Συναρτήσεις - gnvlcode

- μεταφέρει στον t0 την διεύθυνση μιας **μη τοπικής** μεταβλητής
- από τον **πίνακα συμβόλων** βρίσκει πόσα επίπεδα επάνω βρίσκεται η μη τοπική μεταβλητή και μέσα από **τον σύνδεσμο προσπέλασης** την εντοπίζει

```
lw t0,-4(sp)                 στοίβα του γονέα
όσες φορές χρειαστεί:
        lw t0,-4(t0)         στοίβα του προγόνου που έχει τη μεταβλητή
addi t0,t0,-offset           διεύθυνση της μη τοπικής μεταβλητής
```

Βοηθητικές Συναρτήσεις – loadvr

- **μεταφορά δεδομένων στον καταχωρητή r**
- η μεταφορά μπορεί να γίνει από τη μνήμη (στοίβα)
- ή να εκχωρηθεί στο r μία σταθερά
- η σύνταξη της είναι loadvr(v,r)
- διακρίνουμε περιπτώσεις

Βοηθητικές Συναρτήσεις – loadvr

- αν v είναι **σταθερά**

  li tr,v

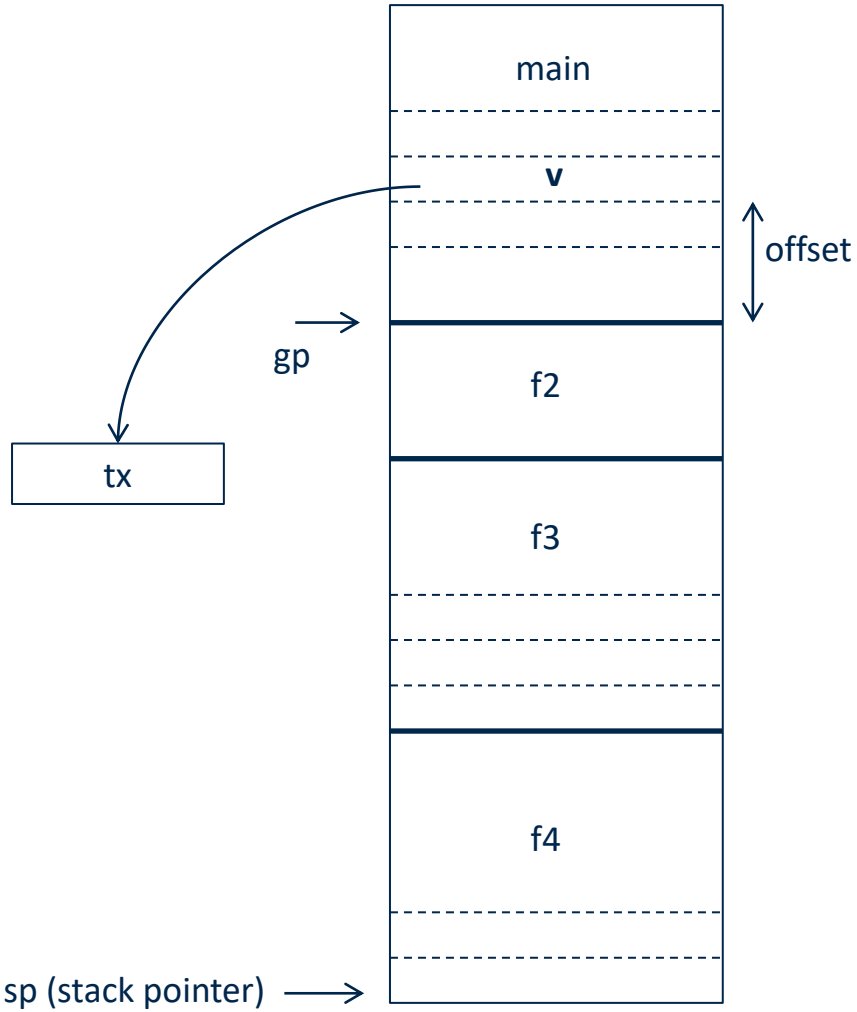
- αν v είναι **καθολική μεταβλητή** – δηλαδή ανήκει στο κυρίως πρόγραμμα

  lw tr,-offset**(gp**)

Βοηθητικές Συναρτήσεις – loadvr

- αν v είναι **καθολική μεταβλητή,** δηλαδή ανήκει στο κυρίως πρόγραμμα

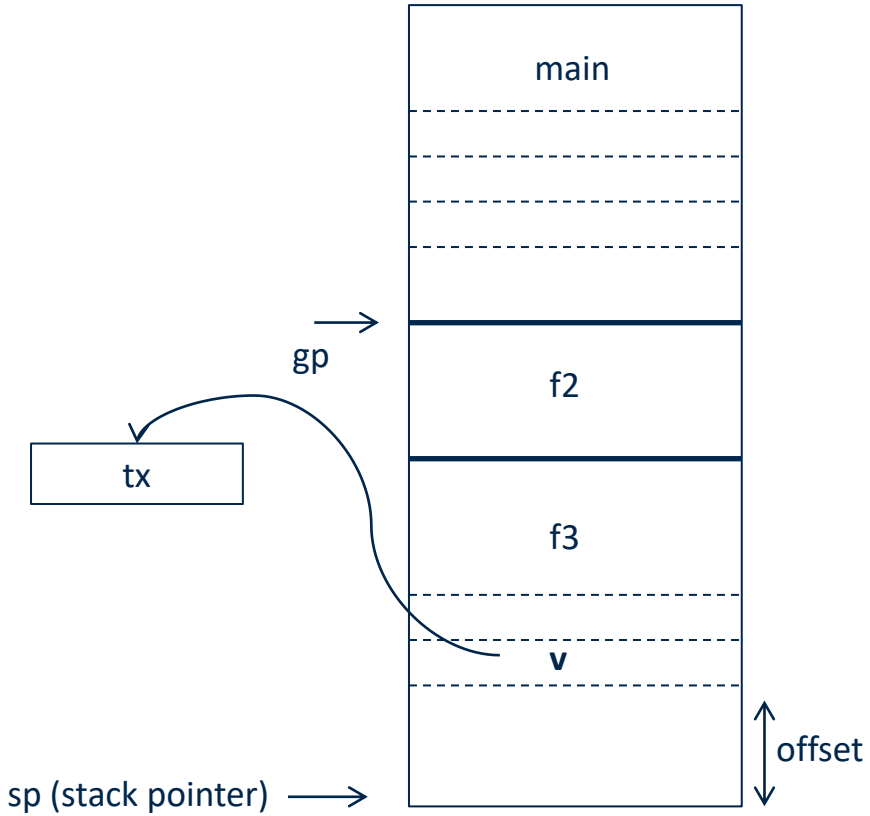
  lw tr,-offset**(gp)**



Βοηθητικές Συναρτήσεις – loadvr

- αν η v έχει δηλωθεί στη συνάρτηση που αυτή τη στιγμή εκτελείται και είναι **τοπική μεταβλητή**, ή **τυπική παράμετρος που περνάει με τιμή**, ή **προσωρινή μεταβλητή**

  lw tr,-offset(sp)

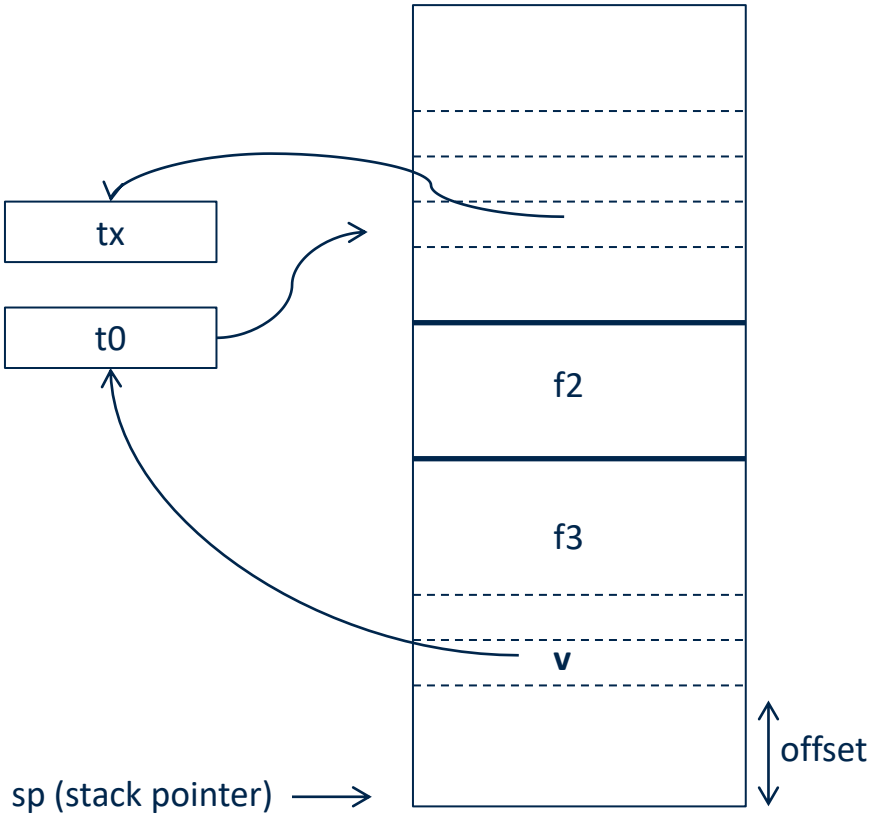


Βοηθητικές Συναρτήσεις – loadvr

- αν η v έχει δηλωθεί στη συνάρτηση που αυτή τη στιγμή εκτελείται και είναι **τυπική παράμετρος που περνάει με αναφορά**

  lw t0,-offset(sp)

  lw tr,(t0)



Βοηθητικές Συναρτήσεις – loadvr

- αν η v έχει δηλωθεί σε κάποιο πρόγονο και εκεί είναι **τοπική μεταβλητή, ή τυπική παράμετρος που περνάει με τιμή**

  gnlvcode()

  lw tr,(t0)

- αν η v έχει δηλωθεί σε κάποιο πρόγονο και εκεί είναι **τυπική παράμετρος που περνάει με αναφορά**

  gnlvcode()

  lw t0,(t0)

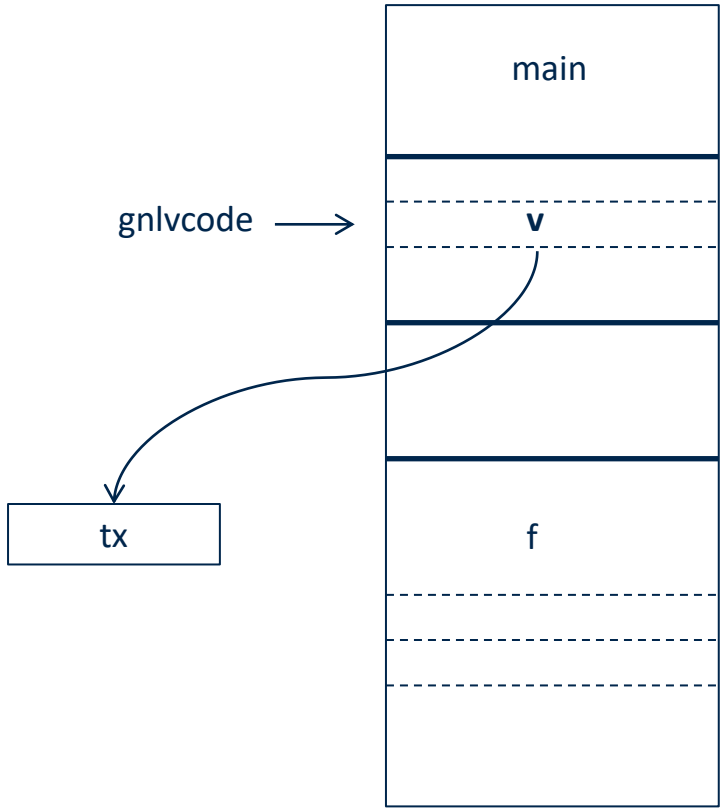
  lw tr,(t0)

Βοηθητικές Συναρτήσεις – loadvr

- αν η v έχει δηλωθεί σε κάποιο πρόγονο και εκεί είναι **τοπική μεταβλητή, ή τυπική παράμετρος που περνάει με τιμή**

gnlvcode()

lw tr,(t0)



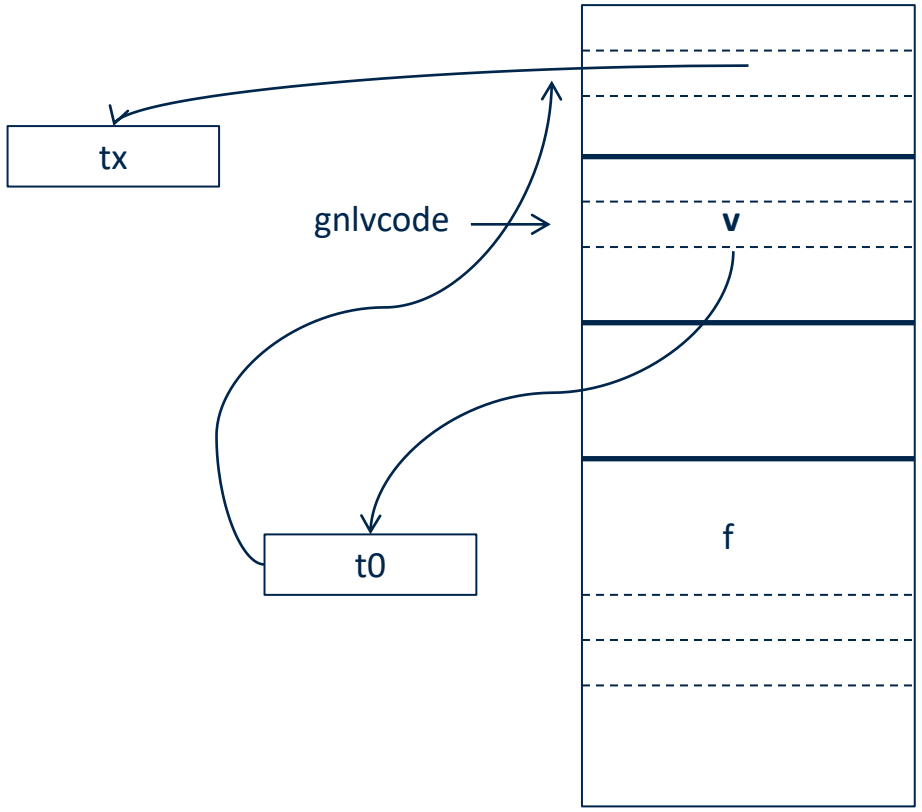
Βοηθητικές Συναρτήσεις – loadvr

- αν η v έχει δηλωθεί σε κάποιο πρόγονο και εκεί είναι **τυπική παράμετρος που περνάει με αναφορά**

  gnlvcode()

  lw t0,(t0)

  lw tr,(t0)



Βοηθητικές Συναρτήσεις – storerv

- **μεταφορά δεδομένων από τον καταχωρητή r στη μνήμη (μεταβλητή v)**
- η σύνταξη της είναι storerv(r,v)
- διακρίνουμε περιπτώσεις

Βοηθητικές Συναρτήσεις – storerv

- αν v είναι **καθολική μεταβλητή** – δηλαδή ανήκει στο κυρίως πρόγραμμα

  sw tr,-offset(gp)

Βοηθητικές Συναρτήσεις – storerv

- αν v είναι **τοπική μεταβλητή, ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος ίσο με το τρέχον, ή προσωρινή μεταβλητή**

```
sw tr,-offset(sp)
```

- αν v είναι **τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος ίσο με το τρέχον**

```
lw t0,-offset(sp)
sw tr,(t0)
```

Βοηθητικές Συναρτήσεις – storerv

- αν v είναι **τοπική μεταβλητή, ή τυπική παράμετρος που περνάει με τιμή και βάθος φωλιάσματος μικρότερο από το τρέχον**

  gnlvcode(v)

  sw tr,(t0)

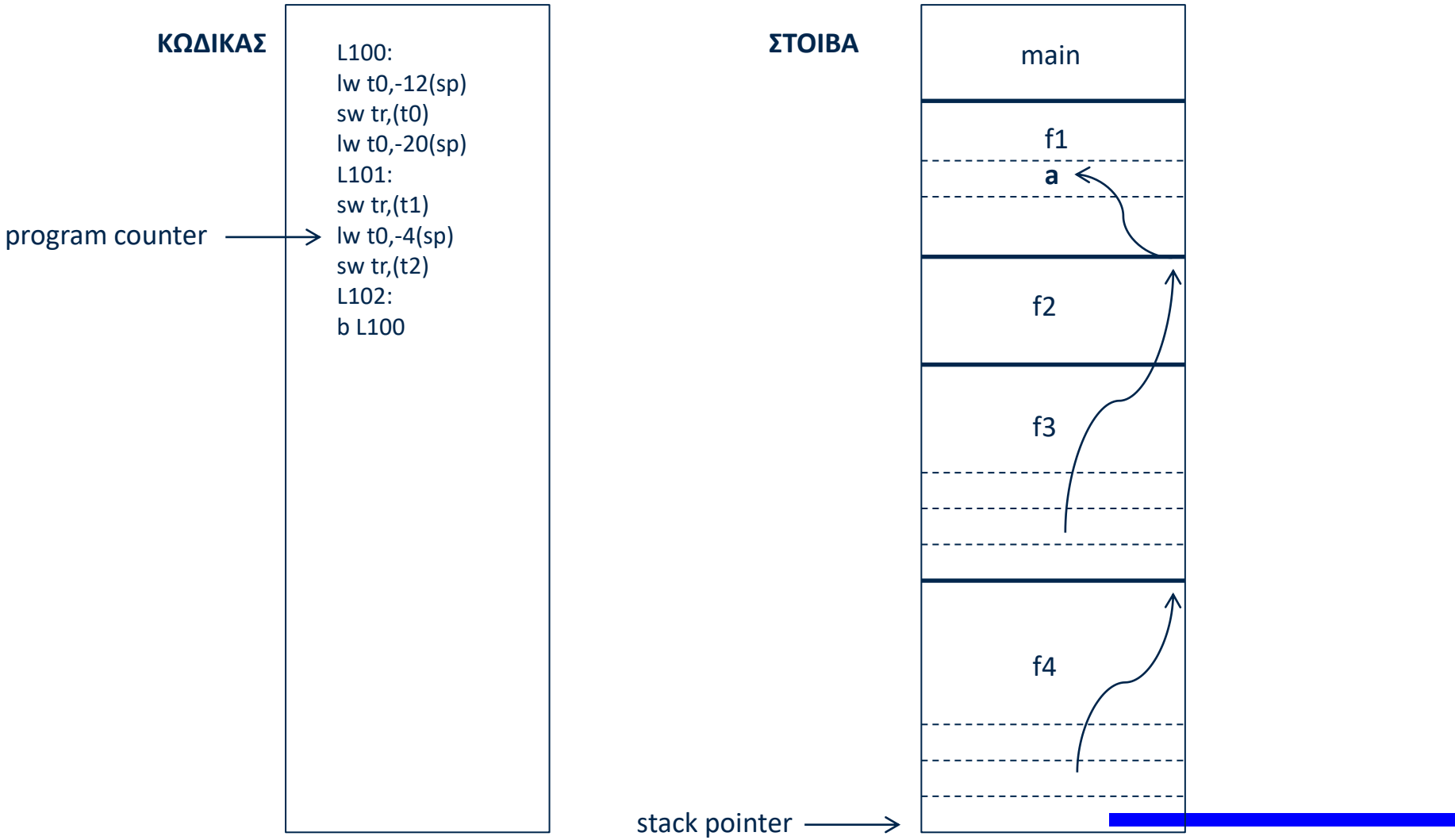
- αν v είναι **τυπική παράμετρος που περνάει με αναφορά και βάθος φωλιάσματος μικρότερο από το τρέχον**

  gnlvcode(v)

  lw t0,(t0)

  sw tr,(t0)

Κώδικας και Δεδομένα



Εντολές Αλμάτων

- **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

Εκχώρηση

- **:=, x, “_”, z**

  loadvr(x, t1)

  storerv(t1, z)

Εντολές Αριθμητικών Πράξεων

- **op x,y,z**

  loadvr(x, t1)

  loadvr(y, t2)

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

  storerv(t1,z)

Επιστροφή Τιμής Συνάρτησης

- **retv “_”, “_”, x**

  loadvr(x, t1)

  lw t0,-8(sp)

  sw t1,(t0)

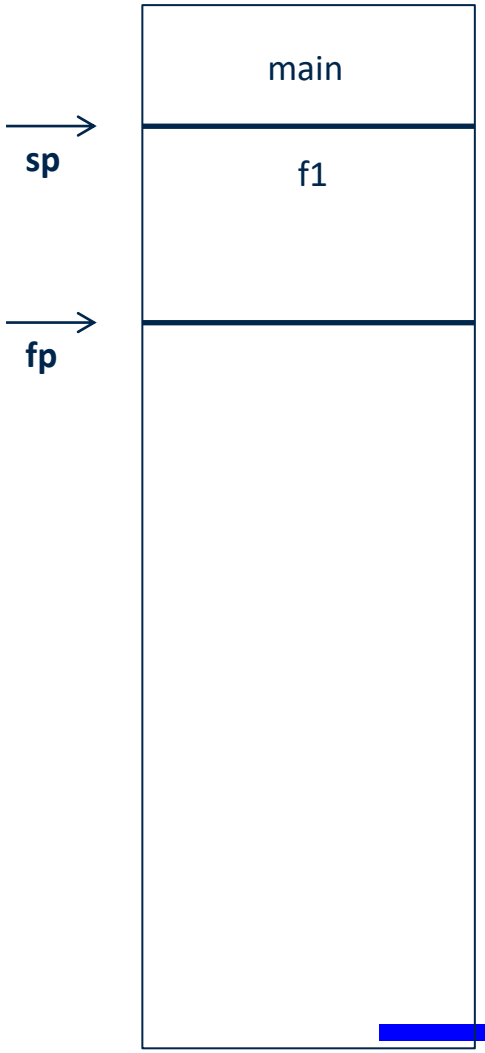
  αποθηκεύεται ο x στη διεύθυνση που είναι αποθηκευμένη στην 3^{η} θέση του εγγραφήματος δραστηριοποίησης

Παράμετροι Συνάρτησης

- πριν κάνουμε τις ενέργειες ώστε να γίνει το **πέρασμα της πρώτης παραμέτρου**, τοποθετούμε τον fp να δείχνει στην στοίβα της συνάρτησης που θα δημιουργηθεί

  addi fp,sp,framelength

- στη συνέχεια, **για κάθε παράμετρο** και ανάλογα με το αν περνά με τιμή ή αναφορά κάνουμε τις εξής ενέργειες:



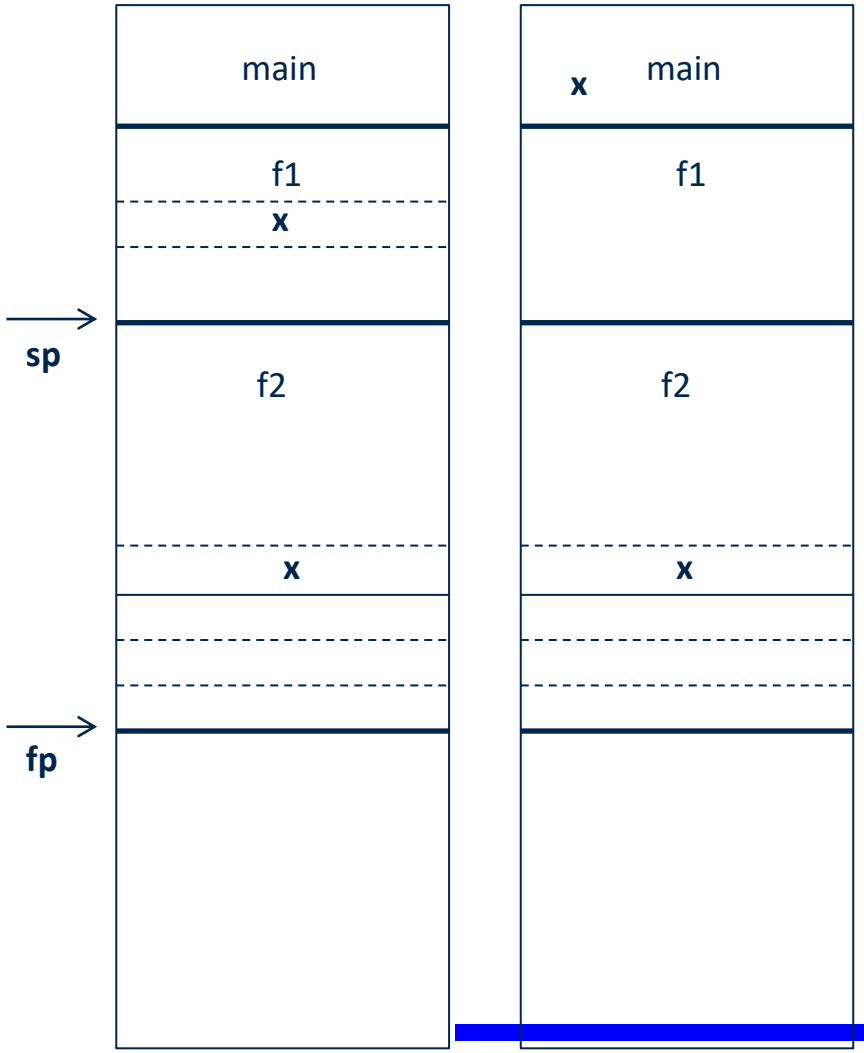
Παράμετροι Συνάρτησης

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

  loadvr(x, t0)

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

  όπου i ο αύξων αριθμός
  της παραμέτρου



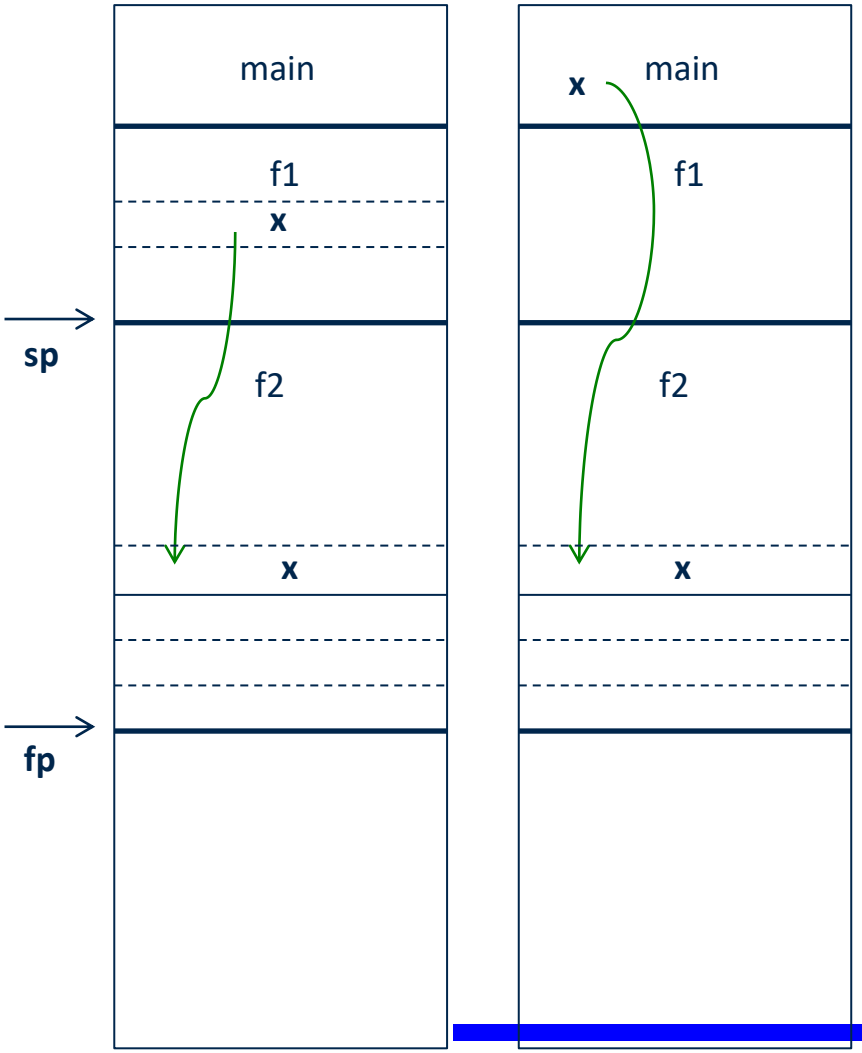
Παράμετροι Συνάρτησης

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

  loadvr(x, t0)

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

  όπου i ο αύξων αριθμός
  της παραμέτρου

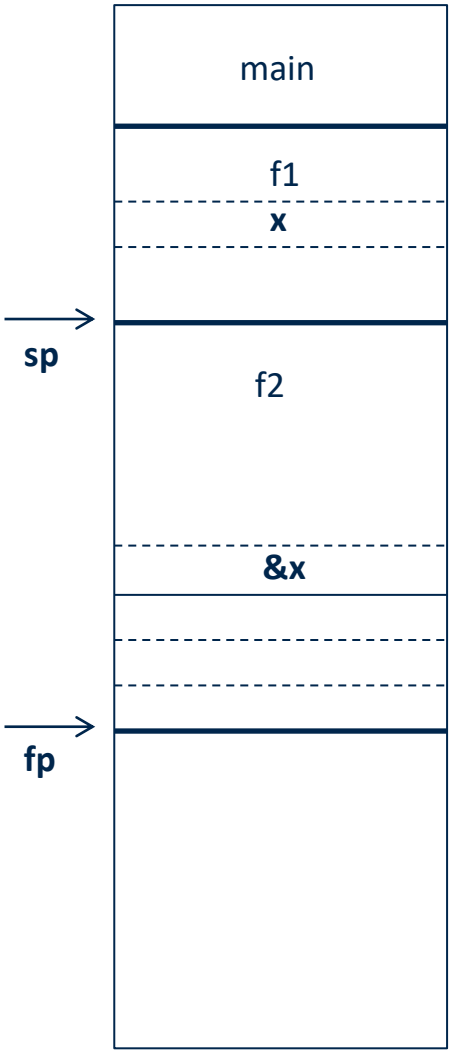


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιο βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    addi t0,sp,-offset

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

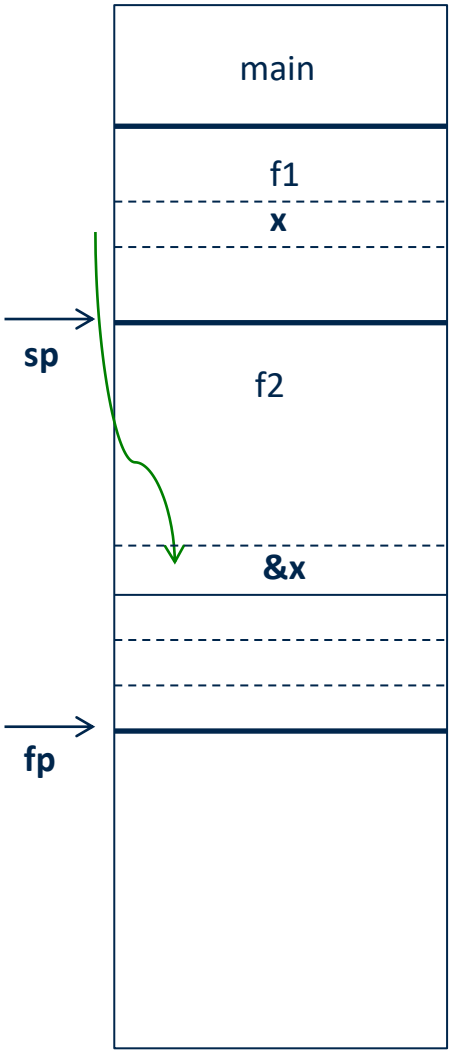


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιο βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    addi t0,sp,-offset

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

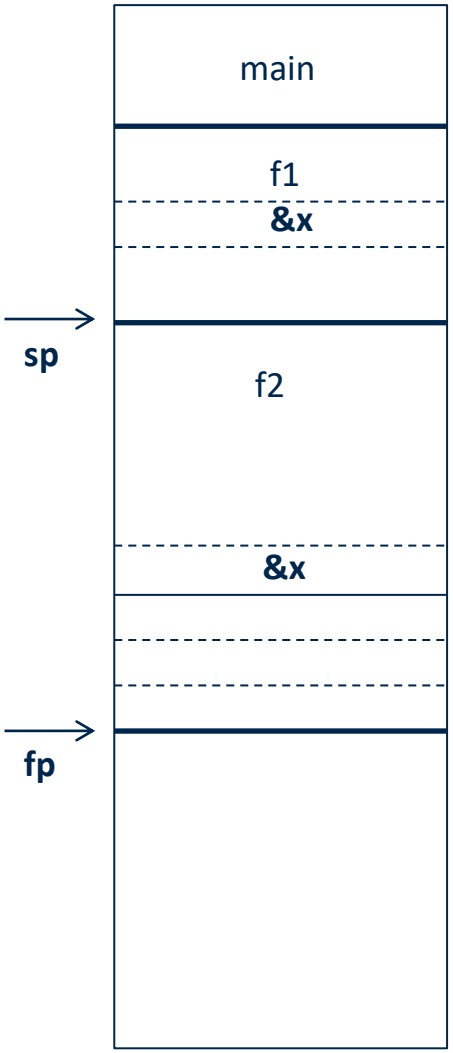


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιο βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    lw t0,-offset(sp)

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

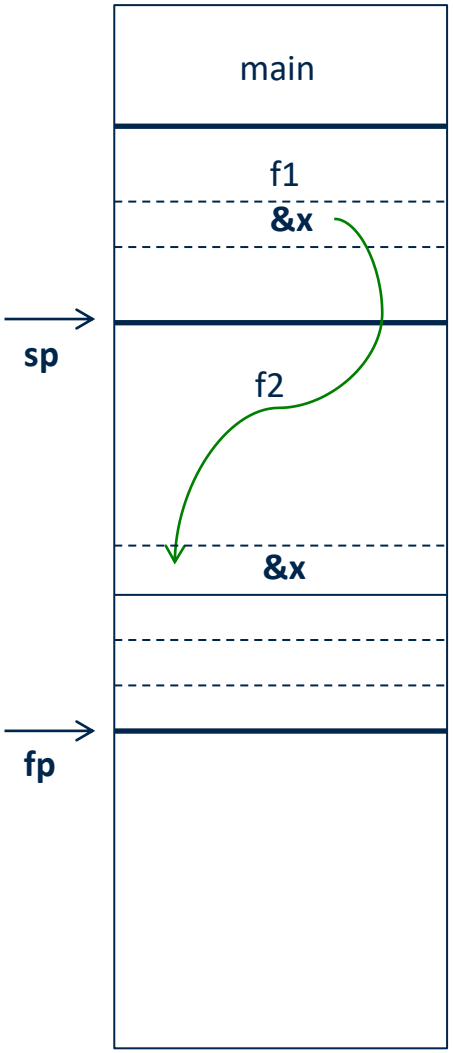


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν το ίδιο βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    lw t0,-offset(sp)

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

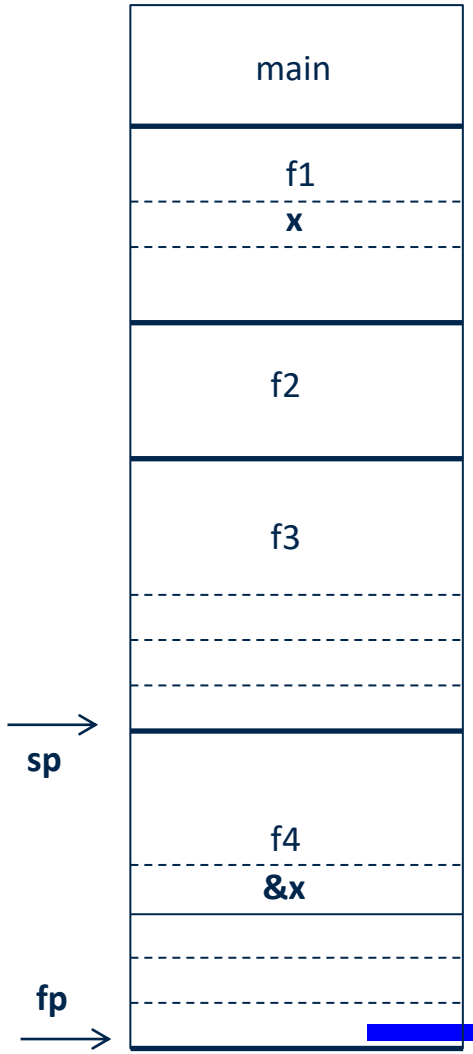


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    gnlvcode(x)

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

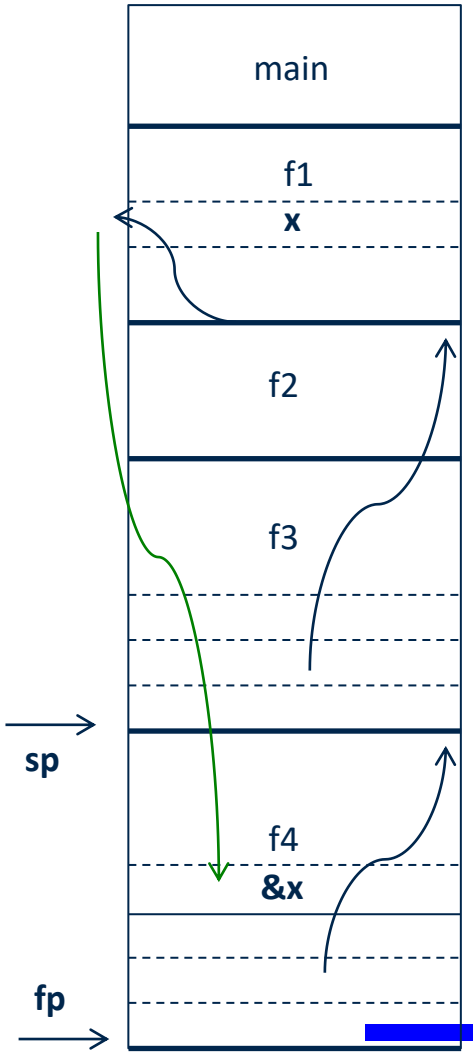


Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση τοπική μεταβλητή ή παράμετρος που έχει περαστεί με τιμή

    gnlvcode(x)

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



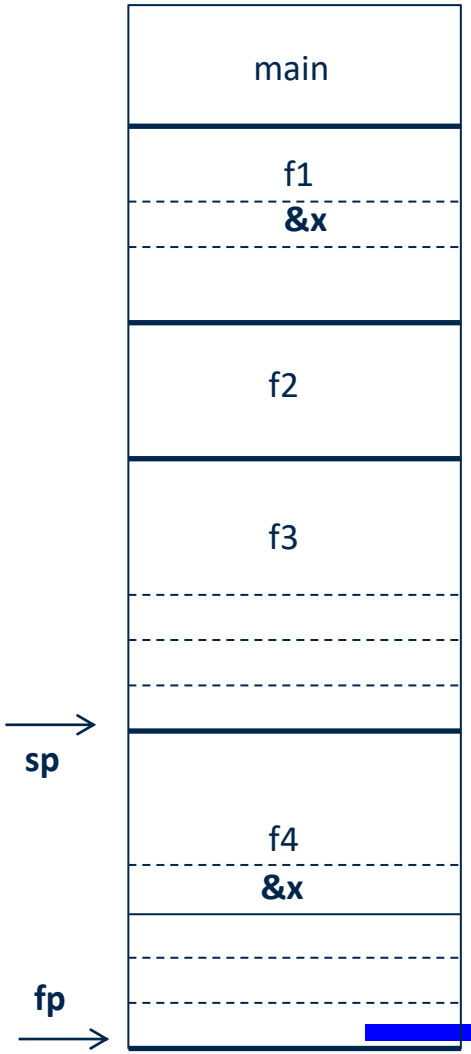
Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    gnlvcode(x)

    lw t0,(t0)

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



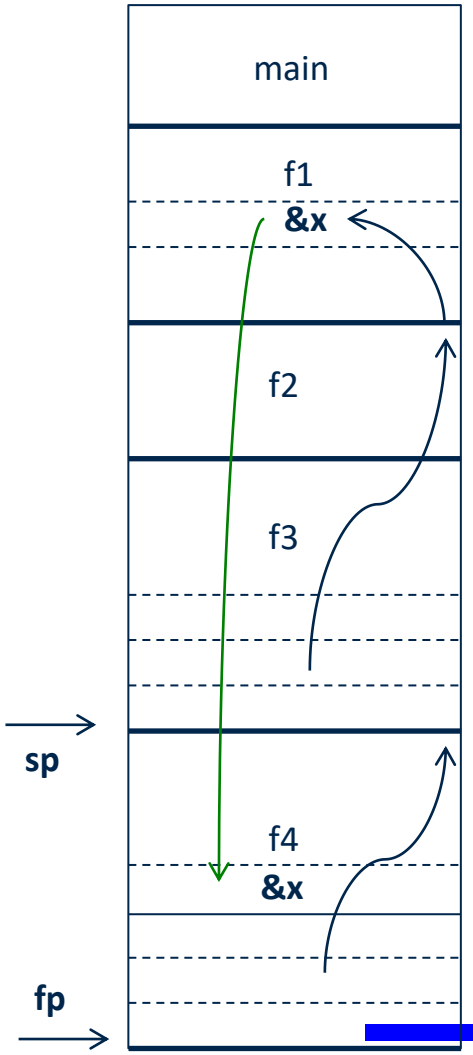
Παράμετροι Συνάρτησης

- **par,x,REF, _**
  - αν η καλούσα συνάρτηση και η μεταβλητή x έχουν διαφορετικό βάθος φωλιάσματος, η παράμετρος x είναι στην καλούσα συνάρτηση παράμετρος που έχει περαστεί με αναφορά

    gnlvcode(x)

    lw t0,(t0)

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



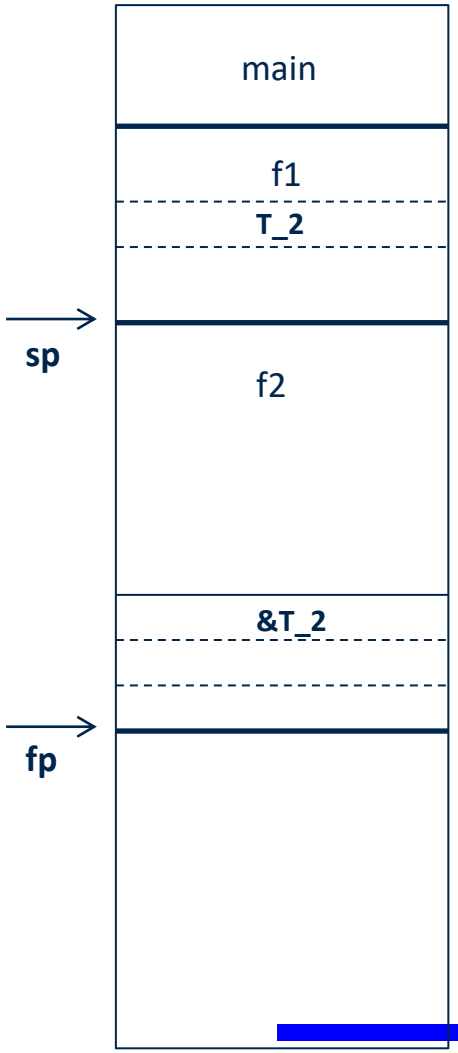
Παράμετροι Συνάρτησης

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

  γεμίζουμε το 3ο πεδίο του εγγραφήματος δραστηριοποίησης της κληθείσας συνάρτησης με τη διεύθυνση της προσωρινής μεταβλητής στην οποία θα επιστραφεί η τιμή

  addi t0,sp,-offset

  sw t0,-8(fp)



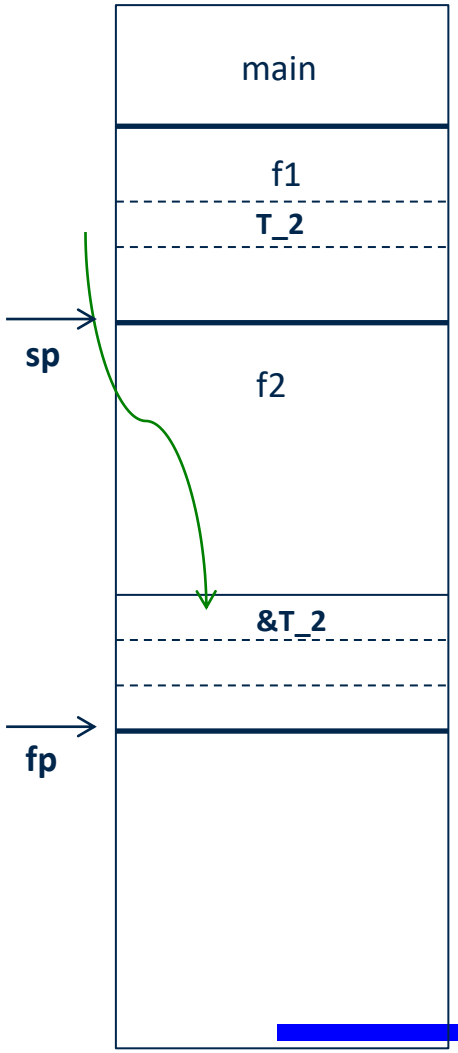
Παράμετροι Συνάρτησης

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

  γεμίζουμε το 3ο πεδίο του εγγραφήματος δραστηριοποίησης της κληθείσας συνάρτησης με τη διεύθυνση της προσωρινής μεταβλητής στην οποία θα επιστραφεί η τιμή

  addi t0,sp,-offset

  sw t0,-8(fp)



Κλήση Συνάρτησης

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

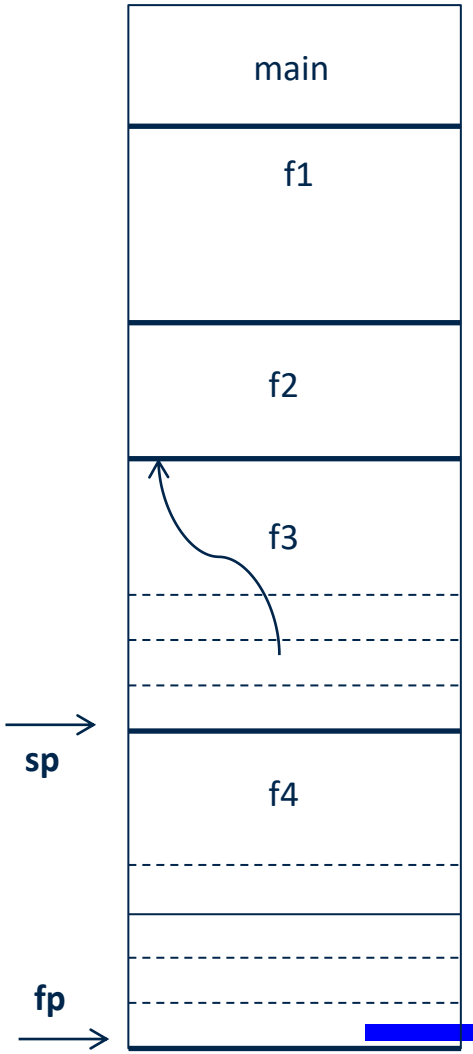
  αρχικά γεμίζουμε το 2ο πεδίο του εγγραφήματος δραστηριοποίησης της κληθείσας συνάρτησης, **τον σύνδεσμο προσπέλασης**, με την διεύθυνση του εγγραφήματος δραστηριοποίησης του γονέα της, ώστε η κληθείσα να γνωρίζει που να κοιτάξει αν χρειαστεί να προσπελάσει μία μεταβλητή την οποία έχει δικαίωμα να προσπελάσει, αλλά δεν της ανήκει

Κλήση Συνάρτησης

- αν καλούσα και κληθείσα έχουν το ίδιο βάθος φωλιάσματος, τότε έχουν τον ίδιο γονέα

  lw t0,-4(sp)

  sw t0,-4(fp)

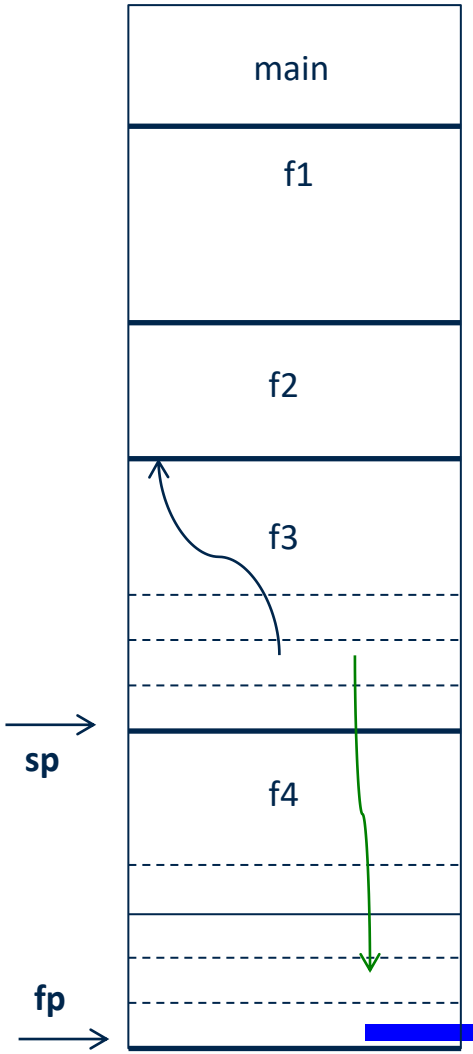


Κλήση Συνάρτησης

- αν καλούσα και κληθείσα έχουν το ίδιο βάθος φωλιάσματος, τότε έχουν τον ίδιο γονέα

  lw t0,-4(sp)

  sw t0,-4(fp)

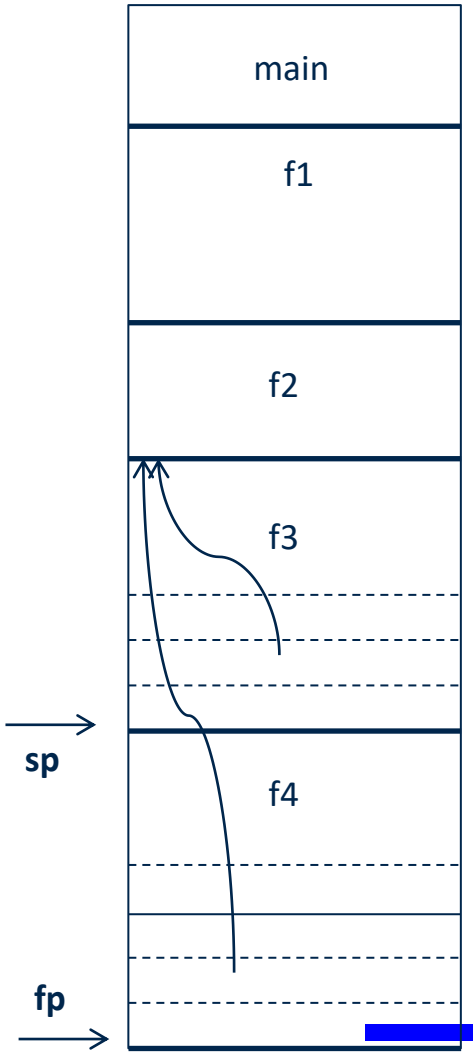


Κλήση Συνάρτησης

- αν καλούσα και κληθείσα έχουν το ίδιο βάθος φωλιάσματος, τότε έχουν τον ίδιο γονέα

  lw t0,-4(sp)

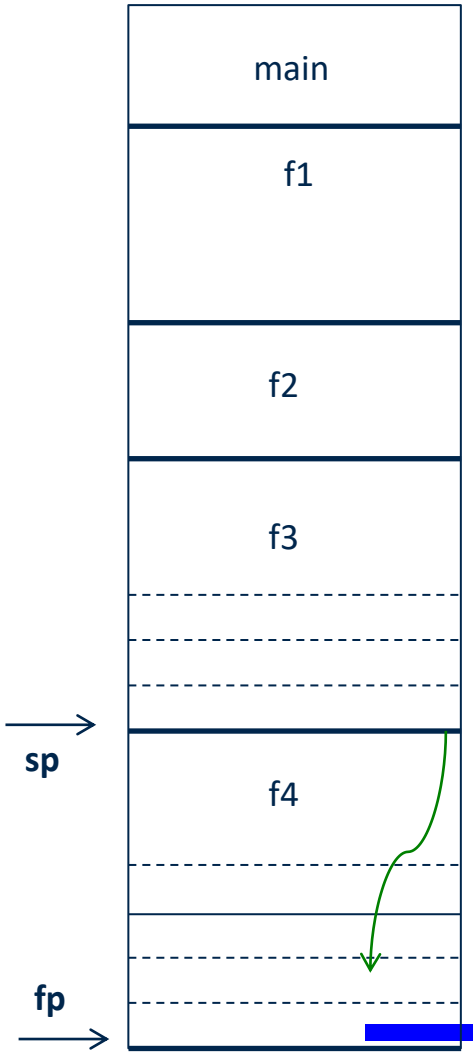
  sw t0,-4(fp)



Κλήση Συνάρτησης

- αν καλούσα και κληθείσα έχουν διαφορετικό βάθος φωλιάσματος, τότε η καλούσα είναι ο γονέας της κληθείσας

  sw sp,-4(fp)



Κλήση Συνάρτησης

- στη συνέχεια **μεταφέρουμε τον δείκτη στοίβας στην κληθείσα**

  addi sp,sp,framelength

- **καλούμε τη συνάρτηση**

  jal f

- και **όταν επιστρέψουμε παίρνουμε πίσω τον δείκτη στοίβας** στην καλούσα

  addi sp,sp,-framelength

Κλήση Συνάρτησης

- **μέσα στην κληθείσα**
  - **στην αρχή** κάθε συνάρτησης αποθηκεύουμε στην πρώτη θέση του εγγραφήματος δραστηριοποίησης την **διεύθυνση επιστροφής** της την οποία έχει τοποθετήσει στον ra η jal

    sw ra,(sp)

  - **στην τέλος** κάθε συνάρτησης κάνουμε το αντίστροφο, παίρνουμε από την πρώτη θέση του εγγραφήματος δραστηριοποίησης την **διεύθυνση επιστροφής** της συνάρτησης και την βάζουμε πάλι στον ra. Μέσω του ra επιστρέφουμε στην καλούσα

    lw ra,(sp)

    jr ra

Αρχή Προγράμματος και Κυρίως Πρόγραμμα

- το κυρίως πρόγραμμα δεν είναι το πρώτο πράγμα που μεταφράζεται, οπότε στην αρχή του προγράμματος χρειάζεται ένα άλμα που να οδηγεί στην πρώτη ετικέτα του κυρίως προγράμματος

  j Lmain

- φυσικά η **j Lmain** πρέπει να δημιουργηθεί όταν ξεκινά η μετάφραση της main
- στη συνέχεια πρέπει να **κατεβάσουμε τον sp κατά framelength της main**

  addi sp,sp,framelength

- και να σημειώσουμε στον gp **το εγγράφημα δραστηριοποίησης της main** ώστε να έχουμε εύκολη πρόσβαση στις global μεταβλητές

  move gp,sp

ευχαριστώ !!!