

Μεταγλωττιστές #6

Νίκος Παπασπύρου

nickie@softlab.ntua.gr



Εθνικό Μετσόβιο Πολυτεχνείο
Τμήμα Ηλεκτρολόγων Μηχ. και Μηχ. Υπολογιστών
Εργαστήριο Τεχνολογίας Λογισμικού
Πολυτεχνειούπολη, 15780 Ζωγράφου.

Απλές εντολές

- Κενή εντολή

$\langle \text{stmt} \rangle ::= \epsilon \{ P_{29} \}$

$P_{29} : \{ \langle \text{stmt} \rangle . \text{NEXT} = \text{EMPTYLIST}(); \}$

- Εντολή ανάθεσης

$\langle \text{stmt} \rangle ::= \langle \text{l-value} \rangle \text{ “:=” } \langle \text{expr} \rangle \{ P_{30} \}$

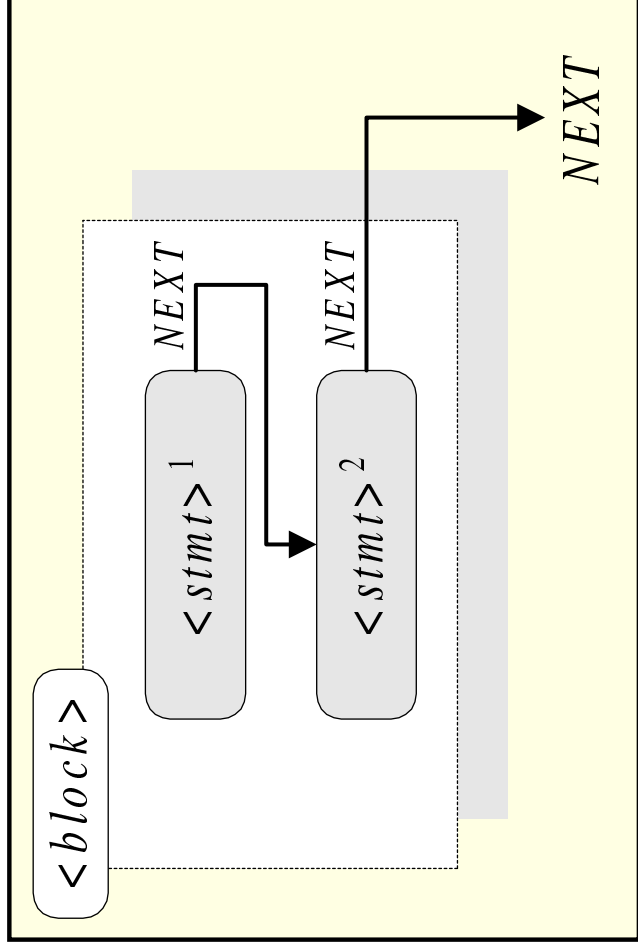
$P_{30} : \{ \text{GENQUAD}(\text{“:=”}, \langle \text{expr} \rangle . \text{PLACE}, -, \langle \text{l-value} \rangle . \text{PLACE}); \langle \text{stmt} \rangle . \text{NEXT} = \text{EMPTYLIST}(); \}$

Σύνθετη εντολή

(i)

$\langle \text{stmt} \rangle ::= \langle \text{block} \rangle$

$\langle \text{block} \rangle ::= \text{“begin” } \langle \text{stmt} \rangle (\text{“,” } \langle \text{stmt} \rangle)^* \text{“end”}$



Σύνθετη εντολή

(ii)

$\langle \text{stmt} \rangle ::= \langle \text{block} \rangle \{ P_{34} \}$

$P_{34} : \{ \langle \text{stmt} \rangle . NEXT = \langle \text{block} \rangle . NEXT; \}$

$\langle \text{block} \rangle ::= \text{“begin”} \langle \text{stmt} \rangle^1 \{ P_{35} \}$
 $(\text{“,”} \{ P_{36} \} \langle \text{stmt} \rangle^2 \{ P_{37} \})^* \text{“end”} \{ P_{38} \}$

$P_{35} : \{ L = \langle \text{stmt} \rangle^1 . NEXT; \}$

$P_{36} : \{ \text{BACKPATCH}(L, \text{NEXTQUAD}()); \}$

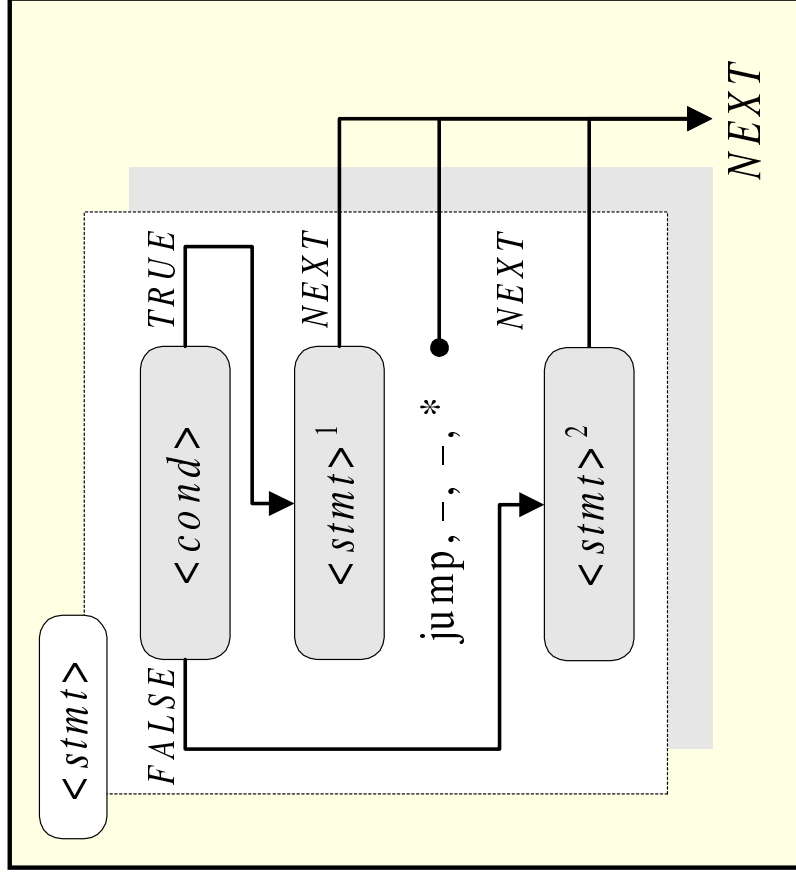
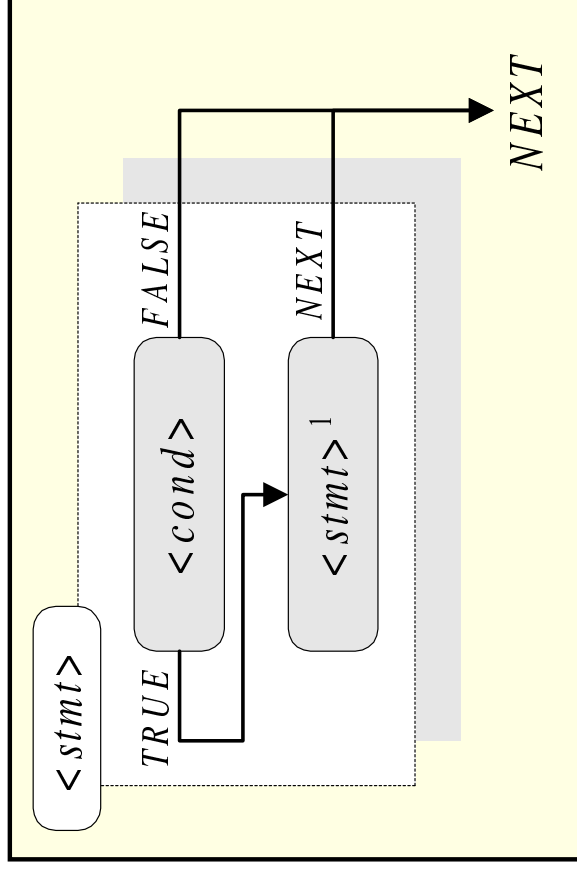
$P_{37} : \{ L = \langle \text{stmt} \rangle^2 . NEXT; \}$

$P_{38} : \{ \langle \text{block} \rangle . NEXT = L; \}$

Εντολή if

(i)

$\langle \text{stmt} \rangle ::= \text{“if”} \langle \text{cond} \rangle \text{“then”} \langle \text{stmt} \rangle [\text{“else”} \langle \text{stmt} \rangle]$



Εντολή if

(ii)

$\langle \text{stmt} \rangle ::= \text{“if” } \langle \text{cond} \rangle \{ P_{39} \} \text{ “then” } \langle \text{stmt} \rangle^1$
[$\text{“else” } \{ P_{40} \} \langle \text{stmt} \rangle^2 \{ P_{41} \}] \{ P_{42} \}$

$P_{39} : \{ \text{BACKPATCH}(\langle \text{cond} \rangle . \text{TRUE}, \text{NEXTQUAD}());$
 $L_1 = \langle \text{cond} \rangle . \text{FALSE};$
 $L_2 = \text{EMPTYLIST}(); \}$

$P_{40} : \{ L_1 = \text{MAKELIST}(\text{NEXTQUAD}());$
 $\text{GENQUAD}(\text{jump}, -, -, *);$
 $\text{BACKPATCH}(\langle \text{cond} \rangle . \text{FALSE}, \text{NEXTQUAD}()); \}$

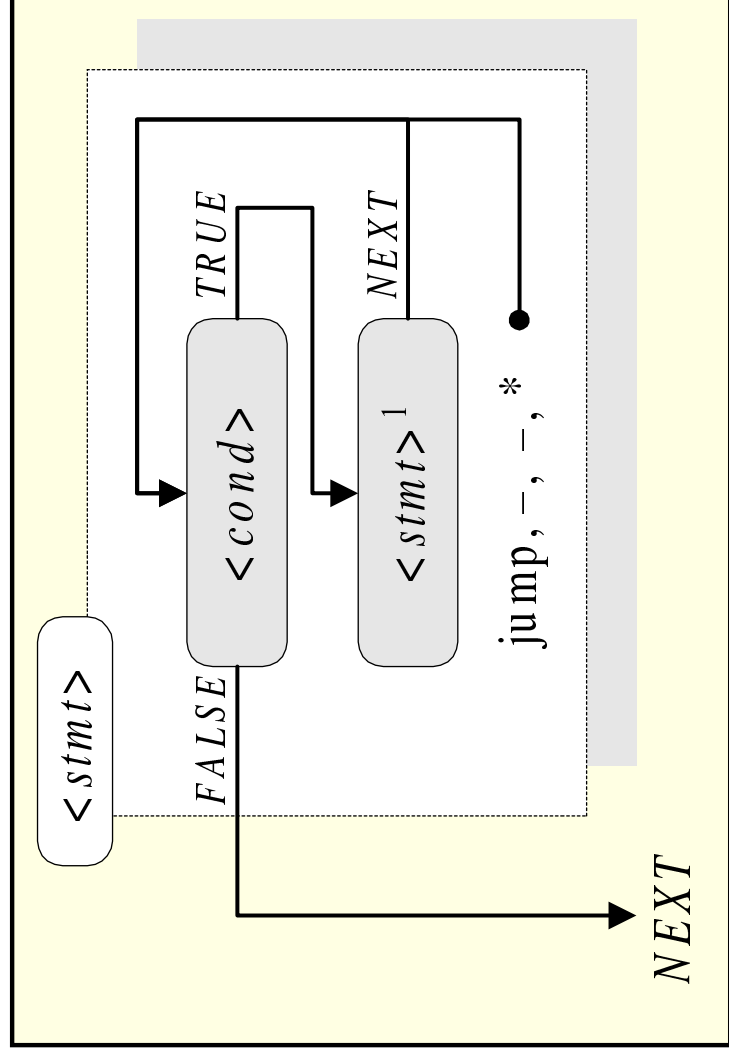
$P_{41} : \{ L_2 = \langle \text{stmt} \rangle^2 . \text{NEXT}; \}$

$P_{42} : \{ \langle \text{stmt} \rangle . \text{NEXT} = \text{MERGE}(L_1, \langle \text{stmt} \rangle^1 . \text{NEXT}, L_2); \}$

Εντολή *while*

(i)

$\langle \text{stmt} \rangle ::= \text{“while”} \langle \text{cond} \rangle \text{“do”} \langle \text{stmt} \rangle$



Εντολή *while*

(ii)

```
 $\langle \text{stmt} \rangle ::= \text{“while” } \{ P_{43} \} \langle \text{cond} \rangle \text{ “do” } \{ P_{44} \} \langle \text{stmt} \rangle^1 \{ P_{45} \}$   
 $P_{43} : \{ Q = \text{NEXTQUAD}(); \}$   
 $P_{44} : \{ \text{BACKPATCH}(\langle \text{cond} \rangle . \text{TRUE}, \text{NEXTQUAD}()); \}$   
 $P_{45} : \{ \text{BACKPATCH}(\langle \text{stmt} \rangle^1 . \text{NEXT}, Q);$   
           $\text{GENQUAD}(\text{jump}, -, -, Q);$   
           $\langle \text{stmt} \rangle . \text{NEXT} = \langle \text{cond} \rangle . \text{FALSE}; \}$ 
```


Κλήση υποπρογραμμαμάτων (i)

$\langle \text{call} \rangle ::= \langle \text{id} \rangle \text{ “(” } [\langle \text{expr} \rangle (\text{ “,” } \langle \text{expr} \rangle)^*] \text{ “)”}$

$\langle \text{r-value} \rangle ::= \langle \text{call} \rangle$

$\langle \text{stmt} \rangle ::= \langle \text{call} \rangle$

- Πέρασμα παραμέτρων με τετράδες par
- Πέρασμα θέσης αποτελέσματος με τετράδα par αν πρόκειται για συνάρτηση
- Κλήση με τετράδα call

Κλήση υποπρογραμμάτων (ii)

$\langle \text{call} \rangle ::= \langle \text{id} \rangle \text{ “(” } \{ P_{46} \} [\langle \text{expr} \rangle^1 \{ P_{47} \}$
 $\quad (\text{ “,” } \langle \text{expr} \rangle^2 \{ P_{48} \}^*] \text{ “)” } \{ P_{49} \}$

$P_{46} : \{ N = 1; \}$

$P_{47} : \{ \text{GENQUAD}(\text{“par”}, \langle \text{expr} \rangle^1.PLACE,$
 $\quad \text{PARAMMODE}(\langle \text{id} \rangle, N), -);$
 $\quad N = N + 1; \}$

$P_{48} : \{ \text{GENQUAD}(\text{“par”}, \langle \text{expr} \rangle^2.PLACE,$
 $\quad \text{PARAMMODE}(\langle \text{id} \rangle, N), -);$
 $\quad N = N + 1; \}$

Κλήση υποπρογραμμάτων (iii)

$\langle \text{call} \rangle ::= \langle \text{id} \rangle \text{ “(” } \{ P_{46} \} [\langle \text{expr} \rangle^1 \{ P_{47} \}$
 $\quad (\text{ “,” } \langle \text{expr} \rangle^2 \{ P_{48} \})^* [\text{ “)” } \{ P_{49} \}$

(συνέχεια)

```
P49 : { if (ISFUNCTION( $\langle \text{id} \rangle$ )) {  
      W = NEWTEMP(FUNCRESULT( $\langle \text{id} \rangle$ ));  
      GENQUAD(par, RET, W, -);  
       $\langle \text{call} \rangle$ .PLACE = W;  
    }  
    GENQUAD(call, -, -,  $\langle \text{id} \rangle$ ); }
```

Κλήση υποπρογραμμάτων (iv)

- Κλήση συνάρτησης

$\langle \text{r-value} \rangle ::= \langle \text{call} \rangle \{ P_{50} \}$

$P_{50} : \{ \langle \text{r-value} \rangle . PLACE = \langle \text{call} \rangle . PLACE; \}$

- Κλήση διαδικασίας

$\langle \text{stmt} \rangle ::= \langle \text{call} \rangle \{ P_{51} \}$

$P_{51} : \{ \langle \text{stmt} \rangle . NEXT = \text{EMPTYLIST}(); \}$

Κλήση υποπρογραμμάτων (v)

- Επιστροφή από υποπρόγραμμα

$\langle \text{stmt} \rangle ::= \text{“return” } [\langle \text{expr} \rangle \{ P_{52} \}] \{ P_{53} \}$

$P_{52} : \{ \text{GENQUAD}(\text{retv}, \langle \text{expr} \rangle.PLACE, -, -); \}$

$P_{53} : \{ \text{GENQUAD}(\text{ret}, -, -, -); \}$

- Δήλωση υποπρογράμματος

$\langle \text{body} \rangle ::= (\langle \text{local} \rangle^* \{ P_{56} \} \langle \text{block} \rangle \text{ “.” } \{ P_{57} \}$

$P_{56} : \{ \text{GENQUAD}(\text{unit}, I, -, -); \}$

$P_{57} : \{ \text{BACKPATCH}(\langle \text{block} \rangle.NEXT, \text{NEXTQUAD}());$
 $\text{GENQUAD}(\text{endu}, I, -, -); \}$

Τελικός κώδικας

(i)

- Από θεωρητικής άποψης, το πρόβλημα της κατασκευής βέλτιστου τελικού κώδικα δεν έχει λύση (undecidable)
- Μορφές τελικού κώδικα:
 - Γλώσσα μηχανής σε απόλυτη μορφή (absolute)
 - Γλώσσα μηχανής σε επανατοποθετήσιμη και διασυνδέσιμη μορφή (relocatable, linkable)
 - Συμβολική γλώσσα (assembly)
 - Άλλη γλώσσα χαμηλού επιπέδου

Τελικός κώδικας

(ii)

- Επιμέρους προβλήματα:

- Επιλογή εντολών

- ⇒ Πώς μεταφράζεται κάθε εντολή του ενδιάμεσου κώδικα
 - ⇒ Πώς μεταφράζονται ακολουθίες τέτοιων εντολών

- Διαχείριση της μνήμης στο χρόνο εκτέλεσης

- ⇒ Πού αποθηκεύονται τα δεδομένα
 - ⇒ Πώς γίνεται η επικοινωνία ανάμεσα στις δομικές μονάδες

Τελικός υπολογιστής

(i)

- Χαρακτηριστικά:
 - Επεξεργαστής: Intel 8086
 - Λειτουργικό σύστημα: MS-DOS
 - Μοντέλο μνήμης: COM / tiny
 - ⇒ Συνολική μνήμη $\leq 64\text{ K}$
 - ⇒ Οργάνωση σε ένα segment
 - ⇒ Αρχική διεύθυνση του προγράμματος η $100h$
 - Συμβολική γλώσσα: συμβατή με το συμβολομεταφραστή MASM (Microsoft macro assembler)

Τελικός υπολογιστής

(ii)

- Καταχωρητές, μεγέθους 16 bit
 - Γενικής φύσης: ax, bx, cx, dx
⇒ σε ζεύγη των 8 bit: ah, al, κ.λπ.
 - Καταχωρητές δείκτες: sp (δείκτης στοίβας) και bp (δείκτης βάσης)
 - Καταχωρητές αναφοράς: si και di
 - Καταχωρητές τμημάτων: cs (code), ds (data), ss (stack) και es (extra)
 - Ειδικό καταχωρητές: ip (instruction pointer) και καταχωρητής σημαίων (flags)

Τελικός υπολογιστής

(iii)

- Διευθύνσεις:

$$address = segment * 16 + offset$$

- Μορφή εντολής:

label *opcode* [*operand*₁ [, *operand*₂]]

Τελικός υπολογιστής

(iv)

- Εντολές:
 - Μεταφοράς: `mov, lea`
 - Αριθμητικών πράξεων: `add, sub, neg, imul, idiv, cmp`
 - Λογικών πράξεων: `and, or, xor, not`
 - Άλματος: `jmp, jz, jnz, jl, jle, jg, jge`
 - Διαχείρισης στοίβας: `push, pop`
 - Υποπρογραμμάτων: `call, ret`

Εντολές μεταφοράς

- *mov destination, source* (move)

```
mov ax, 42
mov ax, bx
mov ax, [1000h]
mov ax, [si]
mov ax, [si + 6]
mov ax, [bp + 6]
mov ax, [si + bp + 6]
```

- *lea destination, source* (load effective address)

- Καθορισμός μεγέθους δεδομένων

```
mov ax, word ptr [bp + 6]
mov al, byte ptr [bp + 6]
```

Αριθμητικές πράξεις

■ `add op_1, op_2`

$op_1 := op_1 + op_2$

■ `sub op_1, op_2`

$op_1 := op_1 - op_2$

■ `neg op`

$op := -op$

■ `imul op`

$(dx, ax) := ax * op$

■ `idiv op`

$ax := (dx, ax) \text{ div } op$

$dx := (dx, ax) \text{ mod } op$

■ `cmp op_1, op_2`

σύγκρινε τα op_1 και op_2
ενημέρωσε τις σημαίες

Λογικές πράξεις

■ **and** op_1, op_2

■ **or** op_1, op_2

■ **not** op

■ **xor** op_1, op_2

$op_1 := op_1$ **and** op_2

$op_1 := op_1$ **or** op_2

$op :=$ **not** op

$op_1 := op_1$ **xor** op_2

Εντολές άλματος

- *jmp address* χωρίς συνθήκη
- *jz address ή je address* μηδέν / ίσο
- *jnz address ή jne address* όχι μηδέν / διάφορο
- *jl address* μικρότερο
- *jle address* μικρότερο ή ίσο
- *jg address* μεγαλύτερο
- *jge address* μεγαλύτερο ή ίσο

Εντολές στοίβας

■ **push** *op*

πρόσθεση στη στοίβα
 $sp := sp - 2, \quad [sp] := op$

■ **pop** *op*

αφαίρεση από τη στοίβα
 $op := [sp], \quad sp := sp + 2$

\Rightarrow Η στοίβα αυξάνει προς τα κάτω, δηλαδή προς μικρότερες διευθύνσεις

Εντολές υποπρογραμμάτων

■ *call address*

κλήση

$sp := sp - 2, \quad [sp] := ip, \quad ip := address$

■ *ret*

επιστροφή

$ip := [sp], \quad sp := sp + 2$

\Rightarrow Η τιμή του ip που τοποθετείται στη στοίβα από την *call* είναι η διεύθυνση της εντολής που ακολουθεί την *call*

Διαχείριση μνήμης

(i)

- Δομή ενότητων (block structure)
 - Μη τοπικά δεδομένα
- Εγγράφημα δραστηριοποίησης (activation record)
 - Παράμετροι
 - Αποτέλεσμα
 - Πληροφορίες κατάστασης μηχανής
 - Τοπικές μεταβλητές
 - Προσωρινές μεταβλητές



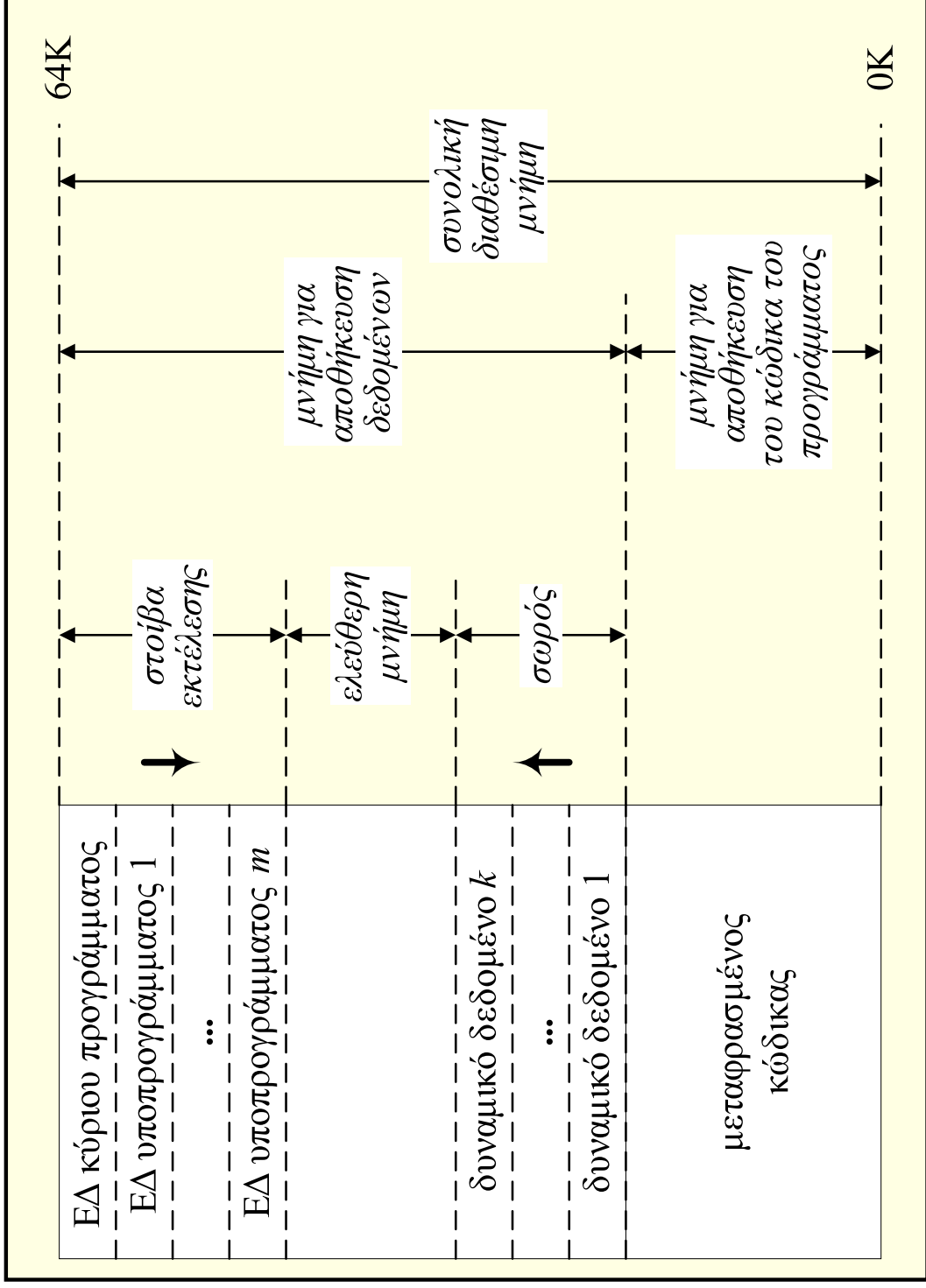
Διαχείριση μνήμης

(ii)

	αρχή			βάση			τέλος
	Παράμετρος 1		Παράμετροι	Παράμετρος 1		Παράμετροι	
	Παράμετρος 2			Παράμετρος 2			
...			
bp+8	Παράμετρος n		Παράμετρος n	Παράμετρος n		Παράμετρος n	
bp+6	Διεύθυνση αποτελέσματος		Διεύθυνση αποτελέσματος	Διεύθυνση αποτελέσματος		Διεύθυνση αποτελέσματος	
bp+4	Σύνδεσμος προσπέλασης		Διεύθυνση επιστροφής	Διεύθυνση επιστροφής		Διεύθυνση επιστροφής	
bp+2	Διεύθυνση επιστροφής		Προηγούμενο bp	Προηγούμενο bp		Προηγούμενο bp	
bp	Προηγούμενο bp		Τοπική μεταβλητή 1	Τοπική μεταβλητή 1		Τοπική μεταβλητή 1	
bp-2	Τοπική μεταβλητή 1		Τοπική μεταβλητή 2	Τοπική μεταβλητή 2		Τοπική μεταβλητή 2	
bp-4	Τοπική μεταβλητή 2		
...	...		Τοπική μεταβλητή m	Τοπική μεταβλητή m		Τοπική μεταβλητή m	
	Προσωρινή μεταβλητή 1		Προσωρινή μεταβλητή 1	Προσωρινή μεταβλητή 1		Προσωρινή μεταβλητή 1	
	Προσωρινή μεταβλητή 2		Προσωρινή μεταβλητή 2	Προσωρινή μεταβλητή 2		Προσωρινή μεταβλητή 2	
	
	Προσωρινή μεταβλητή k		Προσωρινή μεταβλητή k	Προσωρινή μεταβλητή k		Προσωρινή μεταβλητή k	
	α) Σύνδεσμοι προσπέλασης			β) Πίνακας δεικτών			

Διαχείριση μνήμης

(iii)



Προσπέλαση ονομάτων

- Τοπικά: $[bp + offset]$

- Μη τοπικά: $[si + offset]$

\Rightarrow ο si πρέπει να δείχνει στη βάση του εγγραφήματος δραστηριοποίησης όπου τα δεδομένα είναι τοπικά

- Το πρόβλημα ανάγεται στον εντοπισμό του αντίστοιχου εγγραφήματος δραστηριοποίησης

- Λύσεις που βασίζονται στο **βάθος φωλιάσματος**:

- **Σύνδεσμοι προσπέλασης** (access links)
- **Πίνακες δεικτών** (link tables / displays)

Σύνδεσμοι προσπέλασης (i)

- Αρχή λειτουργίας
 - Έστω ότι η δομική μονάδα p βρίσκεται φωλιασμένη μέσα στη δομική μονάδα q
 \Rightarrow Στο $E\Delta$ της p τοποθετείται ένα σύνδεσμος προς τη βάση του $E\Delta$ της πιο πρόσφατης κλήσης της q
- Κατά την κλήση υποπρογραμμάτων, απαιτείται τελικός κώδικας για την ενημέρωση των συνδέσμων προσπέλασης

Σύνδεσμοι προσπέλασης (ii)

■ Τρόπος χρήσης

- Έστω ότι ζητείται το δεδομένο a που είναι τοπικό σε μια δομική μονάδα με βάθος φωλιάσματος n_a
- Έστω ότι βρισκόμαστε σε μια δομική μονάδα p με βάθος φωλιάσματος $n_p \geq n_a$
 \Rightarrow Ακολουθούμε $n_p - n_a$ συνδέσμους προσπέλασης

- Κατά την προσπέλαση ονομάτων, απαιτείται τελικός κώδικας για την υλοποίηση των παραπάνω