Γλώσσα τετράδων

- Μορφή τετράδας:

  n: op, x, y, z

  όπου:

  - n: ετικέτα τετράδας (φυσικός αριθμός)
  - op: τελεστής
  - x, y, z: τελούμενα
- Ανάλογα με το είδος του τελεστή, κάποια τελούμενα ενδεχομένως παραλείπονται

Τελούμενα (i)

- Σταθερά
  - ακέραια, πραγματική, λογική
  - χαρακτήρας, συμβολοσειρά, `nil`
- Όνομα
  - μεταβλητή, παραμέτρος, υποπρόγραμμα
- Προσωρινή μεταβλητή: $\$n$
- Αποτέλεσμα συνάρτησης: $\$\$$
- Αποδεικτοδότηση: $[x]$ x απλό τελούμενο
- Διεύθυνση: $\{x\}$ x απλό τελούμενο

Τελούμενα (ii)

- Ετικέτα
  - εντολής στο αρχικό πρόγραμμα
  - τετράδας
- Τρόπος περάσματος
  - V : κατ' αξία
  - R : κατ' αναφορά
  - RET : θέση αποτελέσματος συνάρτησης
- Κενό : –
- Προσωρινά κενό : * (για backpatching)

Τελεστές (i)

- unit, $I, -, -$
- endu, $I, -, -$
  αρχή και τέλος δομικής μονάδας
- op, x, y, z $\qquad op \in \{+, -, *, /, \%\}$
  $z := x \ op \ y$
- $:=, x, -, z$
  $z := x$
- array, x, y, z
  $z :=$ η διεύθυνση του στοιχείου $x[y]$

Τελεστές (ii)

- op, x, y, z $op \in \{=, <>, >, <, >=, <=\}$
  αν x op y τότε πήγαινε στην τετράδα z
- $\text{ifb}, x, -, z$
  αν η λογική τιμή x είναι αληθής τότε πήγαινε στην τετράδα z
- $\text{jump}, -, -, z$
  πήγαινε στην τετράδα z
- $\text{label}, I, -, -$
  $\text{jumpl}, -, -, I$
  ορισμός ετικέτας και άλμα προς αυτήν

Τελεστές *(iii)*

- $\mathrm{call}, -, -, I$
  κάλεσε τη δομική μονάδα I
- $\mathrm{par}, x, m, -$
  πέρασε την πραγματική παράμετρο x με τρόπο περάσματος m
- $\mathrm{ret}, -, -, -$
  επιστροφή από την τρέχουσα δομική μονάδα

Μεταβλητές ιδιοτήτων

- *PLACE*: θέση όπου βρίσκεται αποθηκευμένη η τιμή μιας l-value ή μιας r-value
- *TYPE*: τύπος μιας l-value ή μιας r-value
- *NEXT*: λίστα από ετικέτες τετράδων που περιέχουν άλματα στην επόμενη εντολή
- *TRUE*, *FALSE*: λίστες από ετικέτες τετράδων που περιέχουν άλματα στον κώδικα που πρέπει να εκτελεστεί αν μια συνθήκη είναι αληθής ή ψευδής

Βοηθητικές υπορουτίνες (i)

- NEXTQUAD()
  Επιστρέφει τον αριθμό της επόμενης τετράδας
- GENQUAD(op, x, y, z)
  Γεννά την επόμενη τετράδα op, x, y, z
- NEWTEMP(t)
  Δημιουργεί μια νέα προσωρινή μεταβλητή τύπου t
- EMPTYLIST()
  Δημιουργεί μια κενή λίστα ετικετών τετράδων

Βοηθητικές υπορουτίνες (ii)

- MAKELIST(x)
  Δημιουργεί μια λίστα ετικετών τετράδων που περιέχει μόνο το στοιχείο x
- MERGE($l_1, \ldots, l_n$)
  Συνένωση των λιστών ετικετών τετράδων $l_1 \ldots l_n$
- BACKPATCH(l, z)
  Αντικαθιστά σε όλες τις τετράδες που περιέχονται στην l την άγνωστη ετικέτα τετράδας με τη z (backpatching)

Αριθμητικές εκφράσεις

- Ακέραιες σταθερές

$$\langle \text{r-value} \rangle \ ::= \ \langle \text{integer-const} \rangle \ \{ P_1 \}$$
$$P_1 \ : \ \{ \ \langle \text{r-value} \rangle . PLACE = \langle \text{integer-const} \rangle ; \ \}$$

Αριθμητικές εκφράσεις

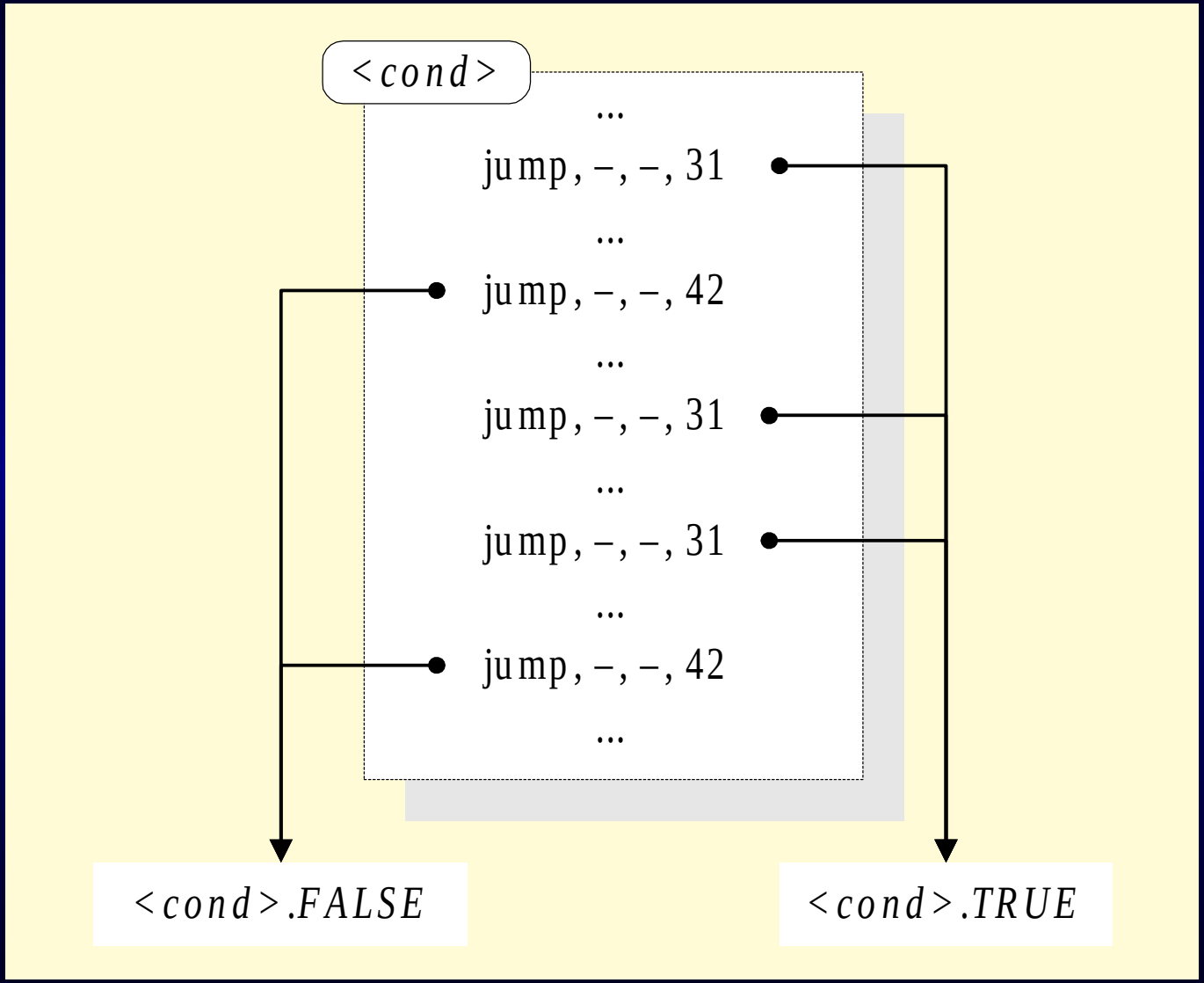
- Ακέραιες σταθερές

$$\langle \text{r-value} \rangle \ ::= \ \langle \text{integer-const} \rangle \ \{ P_1 \}$$
$$P_1 \ : \ \{ \ \langle \text{r-value} \rangle . PLACE = \langle \text{integer-const} \rangle ; \ \}$$

- Τελεστές με δύο τελούμενα

$$\langle \text{r-value} \rangle \ ::= \ \langle \text{expr} \rangle \ \langle \text{binop} \rangle \ \langle \text{expr} \rangle \ \{ P_{14} \}$$
$$P_{14} \ : \ \{ \ W = \text{NEWTEMP}(\langle \text{r-value} \rangle . TYPE);$$
$$\text{GENQUAD}(\langle \text{binop} \rangle . NAME,$$
$$\langle \text{expr} \rangle^1 . PLACE,$$
$$\langle \text{expr} \rangle^2 . PLACE, W);$$
$$\langle \text{r-value} \rangle . PLACE = W; \ \}$$

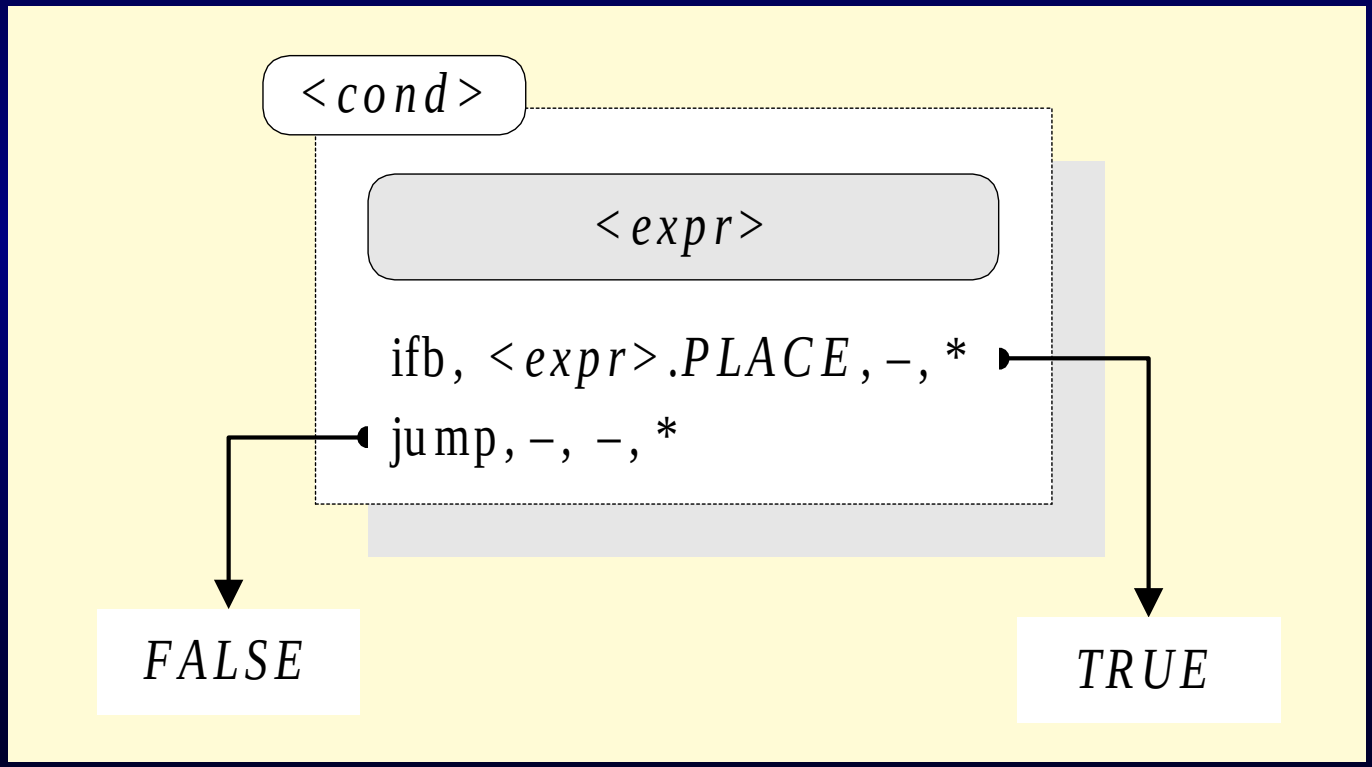
Λογικές εκφράσεις (i)



Λογικές εκφράσεις (ii)

- Λογικές εκφράσεις σε συμβολισμό 0/1

  $\langle \text{cond} \rangle \ ::= \ \langle \text{expr} \rangle$



Λογικές εκφράσεις (iii)

- Λογικές εκφράσεις σε συμβολισμό 0/1

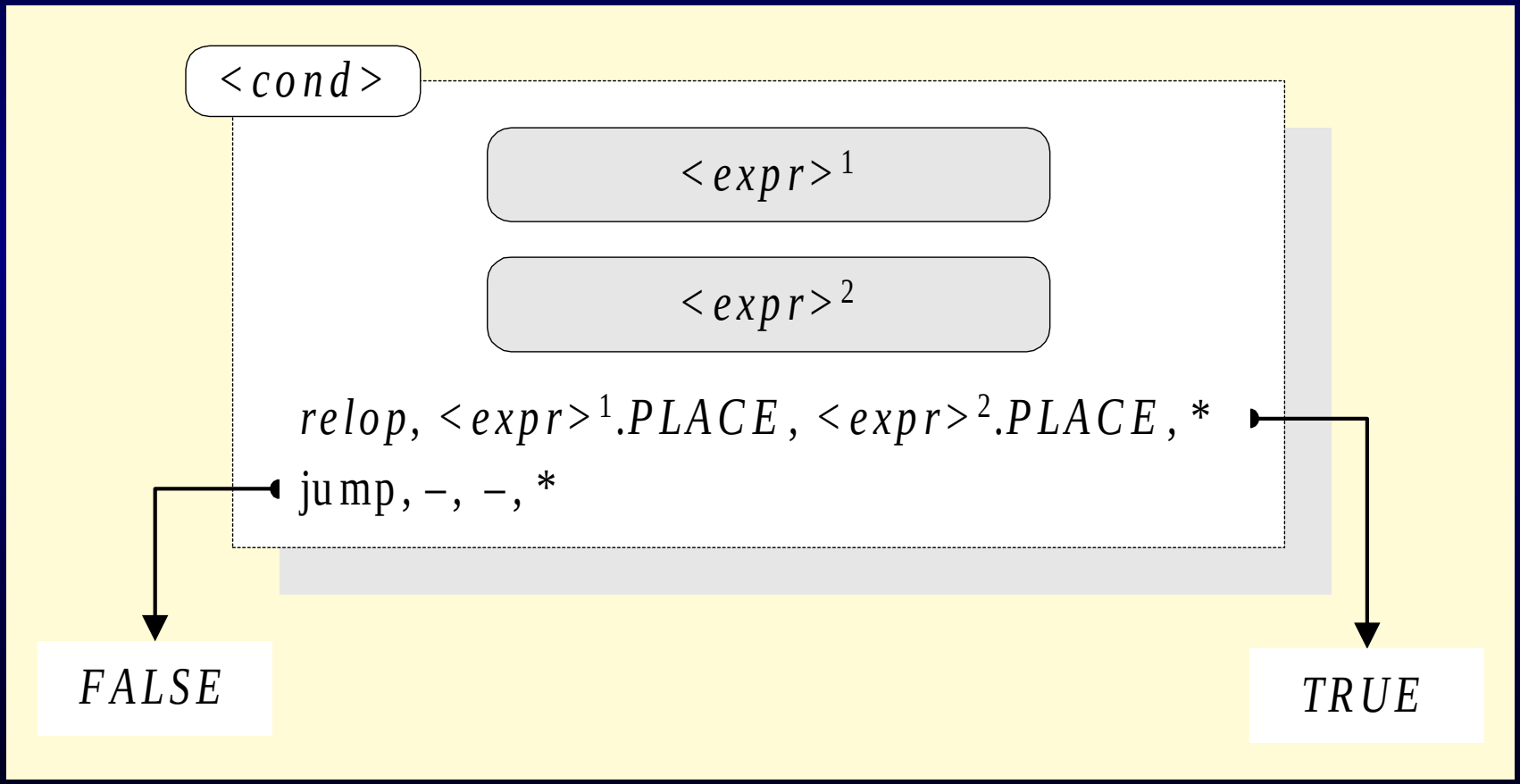
$$\langle \text{cond} \rangle \ ::= \ \langle \text{expr} \rangle \ \{ \, P_{21} \, \}$$

$$P_{21} \ : \ \{ \ \langle \text{cond} \rangle . TRUE = \text{MAKELIST}(\text{NEXTQUAD}());$$
$$\text{GENQUAD}(\text{ifb}, \langle \text{expr} \rangle . PLACE, -, *);$$
$$\langle \text{cond} \rangle . FALSE = \text{MAKELIST}(\text{NEXTQUAD}());$$
$$\text{GENQUAD}(\text{jump}, -, -, *); \ \ \}$$

Λογικές εκφράσεις (iv)

- Τελεστές σύγκρισης

$$\langle \text{cond} \rangle \ ::= \ \langle \text{expr} \rangle^1 \ \langle \text{relop} \rangle \ \langle \text{expr} \rangle^2$$



Λογικές εκφράσεις (v)

- Τελεστές σύγκρισης

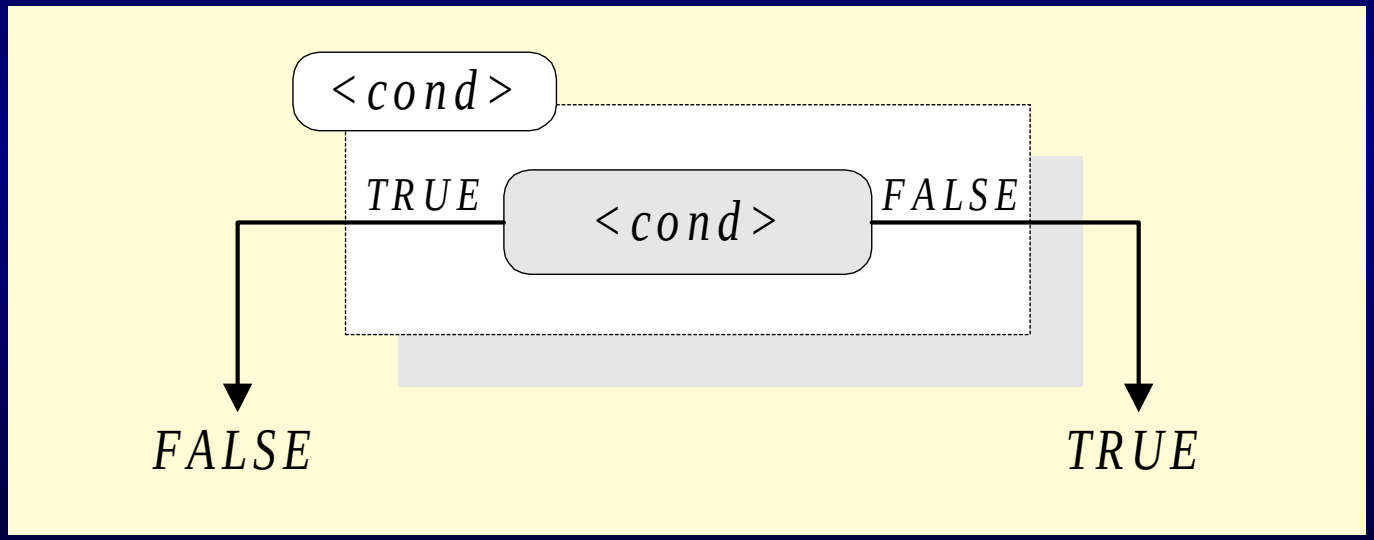
$$\langle \text{cond} \rangle \ ::= \ \langle \text{expr} \rangle^1 \ \langle \text{relop} \rangle \ \langle \text{expr} \rangle^2 \ \{ P_{23} \}$$

$$\begin{aligned} P_{23} \ : \ \{ \ & \langle \text{cond} \rangle.TRUE = \text{MAKELIST}(\text{NEXTQUAD}()); \\ & \text{GENQUAD}(\langle \text{relop} \rangle.NAME, \\ & \qquad \langle \text{expr} \rangle^1.PLACE, \\ & \qquad \langle \text{expr} \rangle^2.PLACE, *); \\ & \langle \text{cond} \rangle.FALSE = \text{MAKELIST}(\text{NEXTQUAD}()); \\ & \text{GENQUAD}(\text{jump}, -, -, *); \ \ \} \end{aligned}$$

Λογικές εκφράσεις (vi)

- Άρνηση

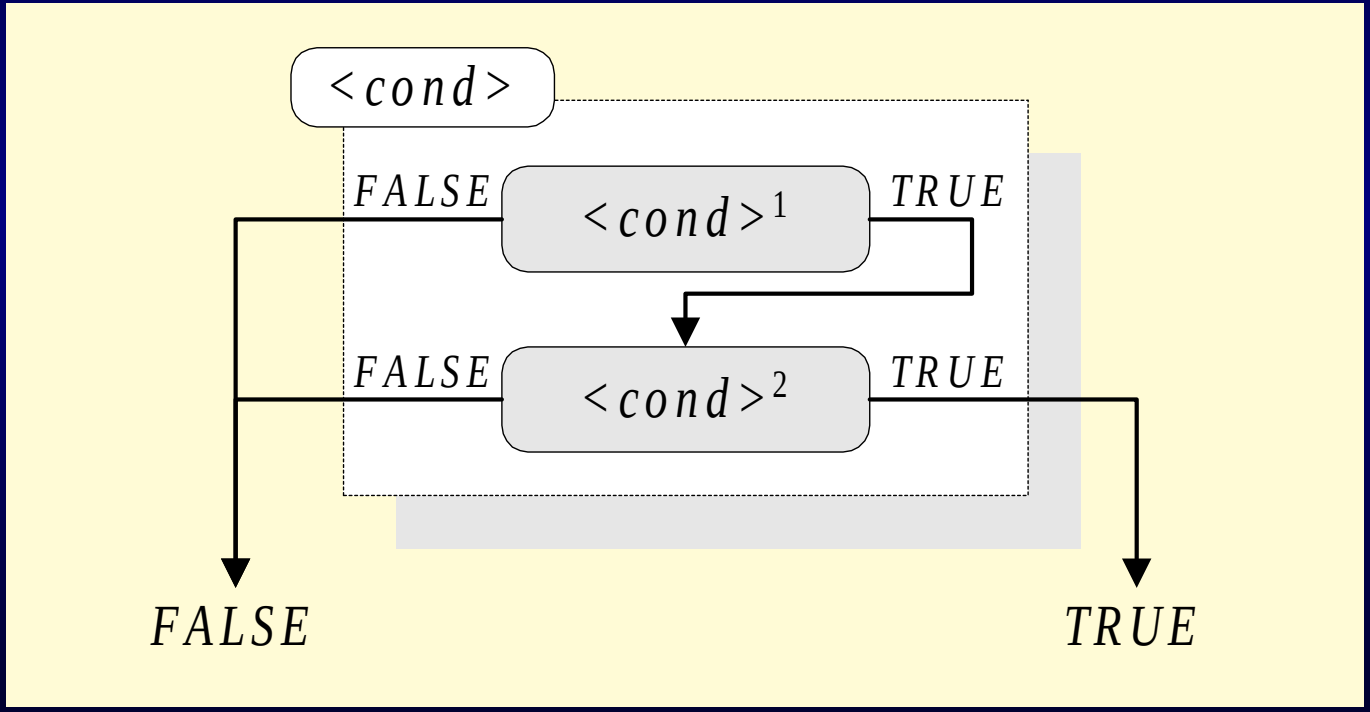
  ⟨cond⟩ ::= “**not**” ⟨cond⟩



Λογικές εκφράσεις (vii)

- Σύζευξη

$\langle \text{cond} \rangle \ ::= \ \langle \text{cond} \rangle^1 \ \text{“}\textbf{and}\text{”} \ \langle \text{cond} \rangle^2$



Λογικές εκφράσεις (viii)

- Σύζευξη

$$\langle \text{cond} \rangle \;::=\; \langle \text{cond} \rangle_1 \;\text{“}\textbf{and}\text{”}\; \{\, P_{25} \,\} \; \langle \text{cond} \rangle_2 \; \{\, P_{26} \,\}$$

$$P_{25} \;:\; \{\; \text{BACKPATCH}(\langle \text{cond} \rangle^1.TRUE, \text{NEXTQUAD}());\; \}$$

$$P_{26} \;:\; \{\; \langle \text{cond} \rangle.FALSE = \text{MERGE}(\langle \text{cond} \rangle^1.FALSE, \langle \text{cond} \rangle^2.FALSE);$$
$$\langle \text{cond} \rangle.TRUE = \langle \text{cond} \rangle^2.TRUE;\; \}$$

Απλές εντολές

- Κενή εντολή

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

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

Απλές εντολές

- Κενή εντολή

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

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

- Εντολή ανάθεσης

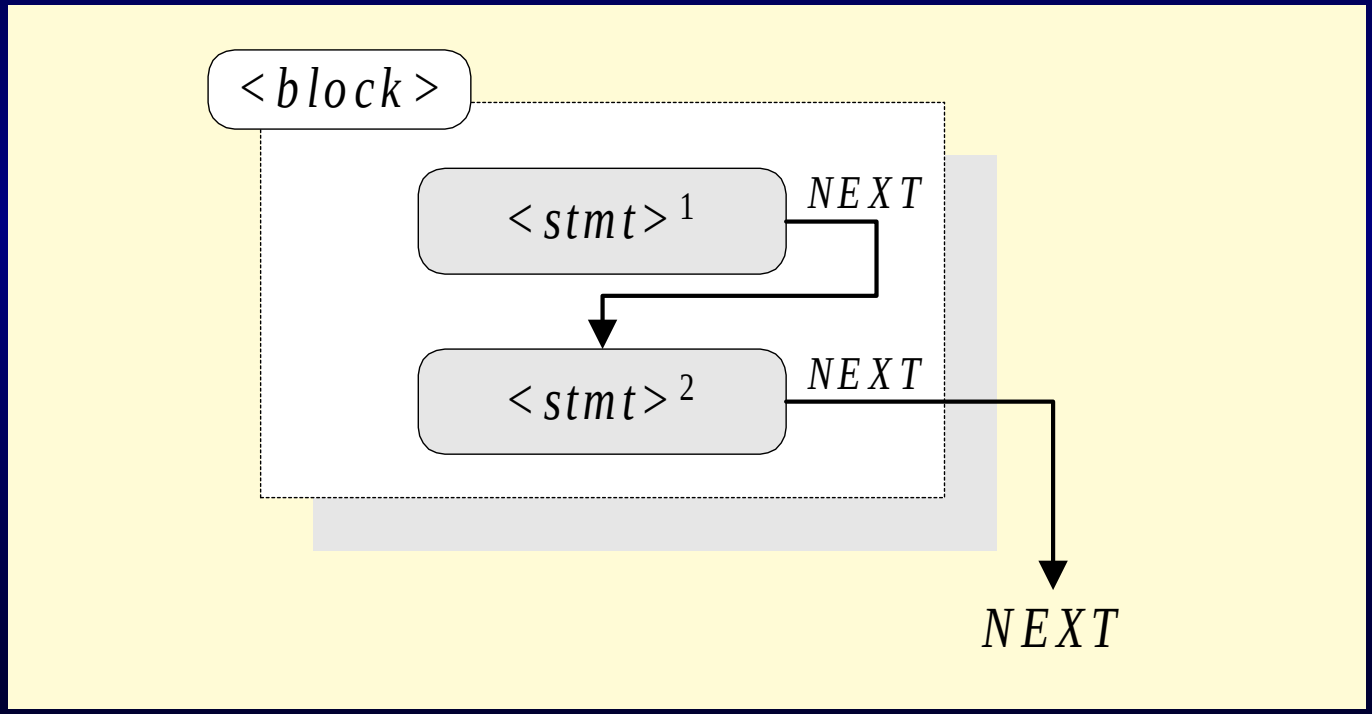
$\langle \text{stmt} \rangle \ ::= \ \langle \text{l-value} \rangle \ \text{“:=”} \ \langle \text{expr} \rangle \ \{ P_{30} \}$

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

Σύνθετη εντολή (i)

⟨stmt⟩ ::= ⟨block⟩

⟨block⟩ ::= “**begin**” ⟨stmt⟩ (“;” ⟨stmt⟩)* “**end**”

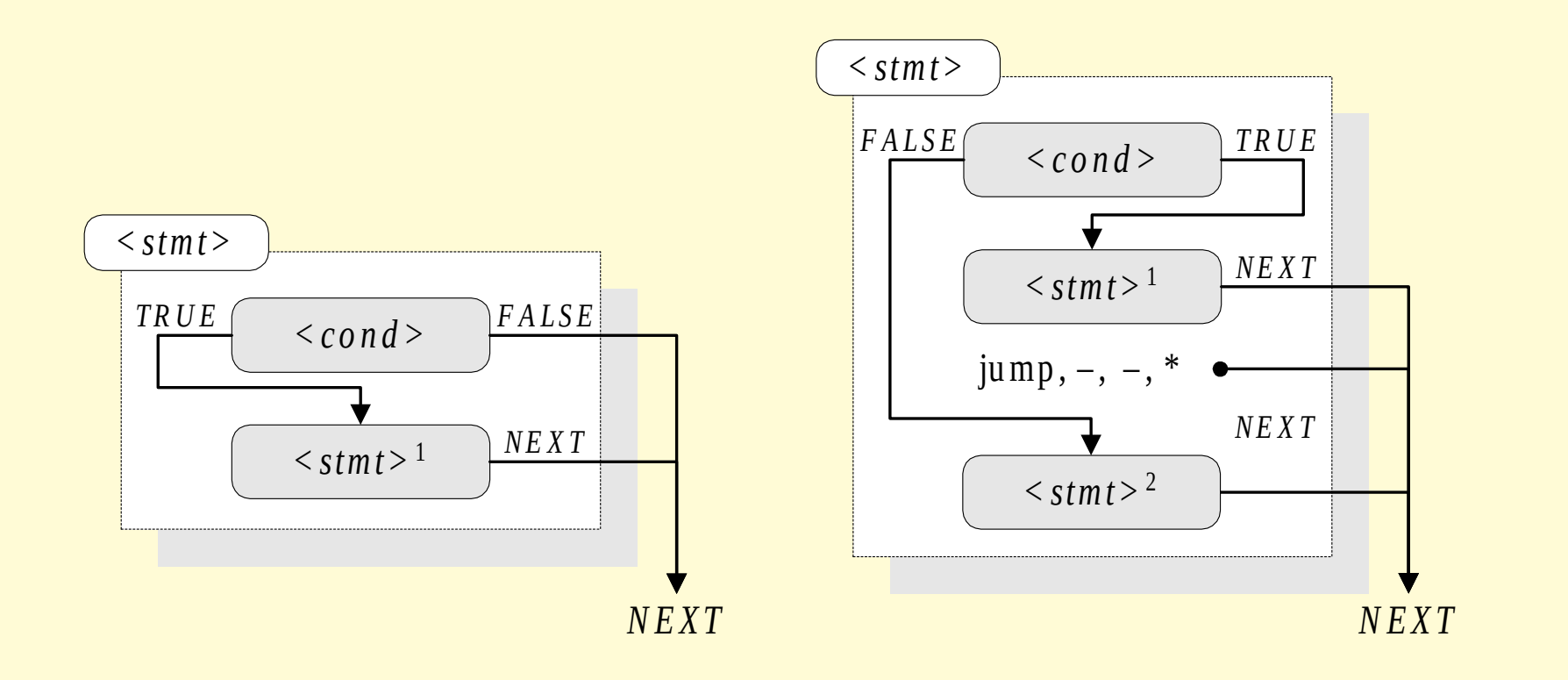


Σύνθετη εντολή (ii)

$$
\begin{array}{lcl}
\langle \text{stmt} \rangle & ::= & \langle \text{block} \rangle \ \{ P_{34} \} \\
P_{34} & : & \{ \ \langle \text{stmt} \rangle.NEXT = \langle \text{block} \rangle.NEXT; \ \} \\
\langle \text{block} \rangle & ::= & \text{“}\textbf{begin}\text{”} \ \langle \text{stmt} \rangle^1 \ \{ P_{35} \} \\
 & & \quad (\ \text{“;”} \ \{ P_{36} \} \ \langle \text{stmt} \rangle^2 \ \{ P_{37} \} \ )^* \ \text{“}\textbf{end}\text{”} \ \{ P_{38} \} \\
P_{35} & : & \{ \ L = \langle \text{stmt} \rangle^1.NEXT; \ \} \\
P_{36} & : & \{ \ \text{BACKPATCH}(L, \text{NEXTQUAD}()); \ \} \\
P_{37} & : & \{ \ L = \langle \text{stmt} \rangle^2.NEXT; \ \} \\
P_{38} & : & \{ \ \langle \text{block} \rangle.NEXT = L; \ \}
\end{array}
$$

Εντολή if (i)

⟨stmt⟩ ::= “**if**” ⟨cond⟩ “**then**” ⟨stmt⟩ [“**else**” ⟨stmt⟩]



Εντολή if (ii)

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

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

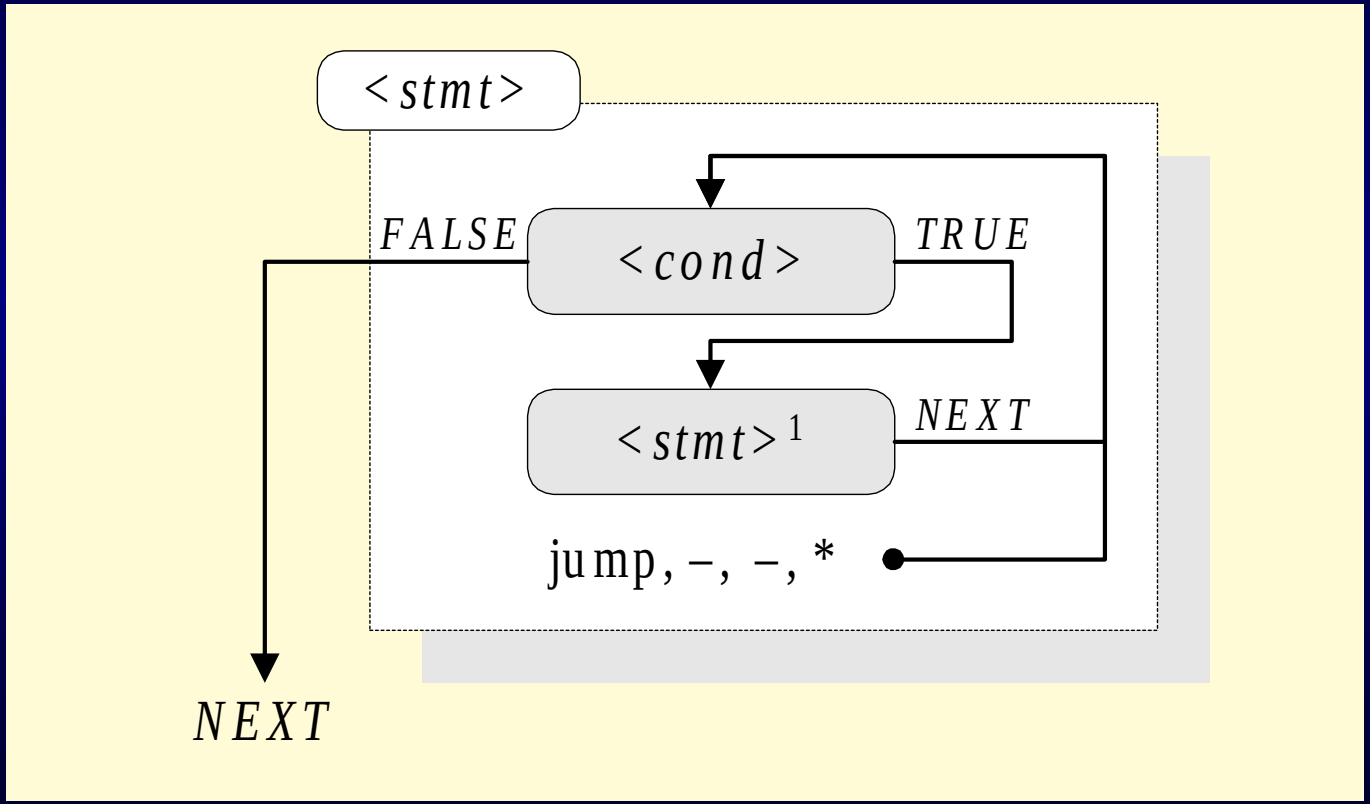
$$P_{40} \ : \ \{ \ L_1 = \text{MAKELIST}(\text{NEXTQUAD}()); \ \text{GENQUAD}(\text{jump}, -, -, *); \ \text{BACKPATCH}(\langle \text{cond} \rangle.FALSE, \text{NEXTQUAD}()); \ \}$$

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

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

Εντολή while (i)

⟨stmt⟩ ::= “**while**” ⟨cond⟩ “**do**” ⟨stmt⟩



Εντολή while (ii)

$$\langle \text{stmt} \rangle \ ::= \ \text{“}\textbf{while}\text{”} \ \{ P_{43} \} \ \langle \text{cond} \rangle \ \text{“}\textbf{do}\text{”} \ \{ P_{44} \} \ \langle \text{stmt} \rangle^1 \ \{ P_{45} \}$$

$$P_{43} \ : \ \{ \ Q = \text{NEXTQUAD}(); \ \}$$

$$P_{44} \ : \ \{ \ \text{BACKPATCH}(\langle \text{cond} \rangle . \mathit{TRUE}, \text{NEXTQUAD}()); \ \}$$

$$P_{45} \ : \ \{ \ \text{BACKPATCH}(\langle \text{stmt} \rangle^1 . \mathit{NEXT}, Q);$$
$$\text{GENQUAD}(\text{jump}, -, -, Q);$$
$$\langle \text{stmt} \rangle . \mathit{NEXT} = \langle \text{cond} \rangle . \mathit{FALSE}; \ \}$$

Κλήση υποπρογραμμάτων (i)

⟨call⟩ ::= ⟨id⟩ “(” [⟨expr⟩ (“,” ⟨expr⟩)*] “)”
⟨r-value⟩ ::= ⟨call⟩
⟨stmt⟩ ::= ⟨call⟩

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

Κλήση υποπρογραμμάτων (ii)

$$
\begin{aligned}
\langle \text{call} \rangle \;::=\; & \langle \text{id} \rangle \;\text{“(”}\; \{\, P_{46} \,\}\; [\; \langle \text{expr} \rangle^1 \;\{\, P_{47} \,\} \\
& (\; \text{“,”}\; \langle \text{expr} \rangle^2 \;\{\, P_{48} \,\}\;)^*\;]\; \text{“)”}\; \{\, P_{49} \,\} \\
P_{46} \;:\; & \{\; N = 1; \;\} \\
P_{47} \;:\; & \{\; \text{GENQUAD}(\text{“}\mathbf{par}\text{”}, \langle \text{expr} \rangle^1.PLACE, \\
& \qquad \text{PARAMMODE}(\langle \text{id} \rangle, N), -); \\
& \quad N = N + 1; \;\} \\
P_{48} \;:\; & \{\; \text{GENQUAD}(\text{“}\mathbf{par}\text{”}, \langle \text{expr} \rangle^2.PLACE, \\
& \qquad \text{PARAMMODE}(\langle \text{id} \rangle, N), -); \\
& \quad N = N + 1; \;\}
\end{aligned}
$$

Κλήση υποπρογραμμάτων (iii)

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

(συνέχεια)

P_{49} : { **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 υποπρογραμμάτων (iv)

- Κλήση συνάρτησης

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

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

- Κλήση διαδικασίας

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

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

Κλήση υποπρογραμμάτων (v)

- Επιστροφή από υποπρόγραμμα

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

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

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

Κλήση υποπρογραμμάτων (v)

- Επιστροφή από υποπρόγραμμα

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

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

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

- Δήλωση υποπρογράμματος

  $\langle \text{body} \rangle$ ::= ($\langle \text{local} \rangle$)* { P_{56} } $\langle \text{block} \rangle$ ";" { P_{57} }

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

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

Παράδειγμα (i)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
  while i <= j do begin
    while a[i] < a[(m+n) div 2] do i := i+1;
    while a[j] > a[(m+n) div 2] do j := j-1;
    if i <= j then begin
      temp := a[i]; a[i] := a[j]; a[j] := temp;
      i := i+1; j := j-1
    end
  end;
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, *

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, *
3: jump, −, −, *

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, *
3: jump, −, −, *
4: ret, −, −, −

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, 4
3: jump, −, −, *
4: ret, −, −, −

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, 4
3: jump, −, −, 5
4: ret, −, −, −

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, 4
3: jump, −, −, 5
4: ret, −, −, −
5: :=, m, −, i

Παράδειγμα (ii)

```
procedure quicksort (var a : array of integer;
                     m, n : integer);
   var i, j, temp : integer;
begin
  if n <= m then return;
  i := m; j := n;
```

1: unit, quicksort, −, −
2: <=, n, m, 4
3: jump, −, −, 5
4: ret, −, −, −
5: :=, m, −, i
6: :=, n, −, j

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=$, i, j, $*$

8: jump, $-$, $-$, $*$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$

8: $\mathrm{jump}, -, -, *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=$, i, j, 9
8: jump, $-$, $-$, $*$
9: array, a, i, \$1
10: $+$, m, n, \$2

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], *$
14: $\mathrm{jump}, -, -, *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, *$
15: $+, \mathrm{i}, 1, \$5$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, *$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, *$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$
22: $>, [\$6], [\$9], *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$
22: $>, [\$6], [\$9], *$
23: $\mathrm{jump}, -, -, *$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=$, i, j, 9
8: jump, $-$, $-$, $*$
9: array, a, i, \$1
10: $+$, m, n, \$2
11: $/$, \$2, 2, \$3
12: array, a, \$3, \$4
13: $<$, [\$1], [\$4], 15
14: jump, $-$, $-$, 18
15: $+$, i, 1, \$5
16: $:=$, \$5, $-$, i
17: jump, $-$, $-$, 9
18: array, a, j, \$6
19: $+$, m, n, \$7
20: $/$, \$7, 2, \$8
21: array, a, \$8, \$9
22: $>$, [\$6], [\$9], 24
23: jump, $-$, $-$, $*$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \text{i}, \text{j}, 9$
8: $\text{jump}, -, -, *$
9: $\text{array}, \text{a}, \text{i}, \$1$
10: $+, \text{m}, \text{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\text{array}, \text{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\text{jump}, -, -, 18$
15: $+, \text{i}, 1, \$5$
16: $:=, \$5, -, \text{i}$
17: $\text{jump}, -, -, 9$
18: $\text{array}, \text{a}, \text{j}, \$6$
19: $+, \text{m}, \text{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\text{array}, \text{a}, \$8, \$9$
22: $>, [\$6], [\$9], 24$
23: $\text{jump}, -, -, *$
24: $-, \text{j}, 1, \$10$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$
22: $>, [\$6], [\$9], 24$
23: $\mathrm{jump}, -, -, *$
24: $-, \mathrm{j}, 1, \$10$
25: $:=, \$10, -, \mathrm{j}$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$
22: $>, [\$6], [\$9], 24$
23: $\mathrm{jump}, -, -, *$
24: $-, \mathrm{j}, 1, \$10$
25: $:=, \$10, -, \mathrm{j}$
26: $\mathrm{jump}, -, -, 18$

Παράδειγμα (iii)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
9: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$1$
10: $+, \mathrm{m}, \mathrm{n}, \$2$
11: $/, \$2, 2, \$3$
12: $\mathrm{array}, \mathrm{a}, \$3, \$4$
13: $<, [\$1], [\$4], 15$
14: $\mathrm{jump}, -, -, 18$
15: $+, \mathrm{i}, 1, \$5$
16: $:=, \$5, -, \mathrm{i}$
17: $\mathrm{jump}, -, -, 9$
18: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$6$
19: $+, \mathrm{m}, \mathrm{n}, \$7$
20: $/, \$7, 2, \$8$
21: $\mathrm{array}, \mathrm{a}, \$8, \$9$
22: $>, [\$6], [\$9], 24$
23: $\mathrm{jump}, -, -, 27$
24: $-, \mathrm{j}, 1, \$10$
25: $:=, \$10, -, \mathrm{j}$
26: $\mathrm{jump}, -, -, 18$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, *$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, *$
28: $\mathrm{jump}, -, -, *$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$

28: $\mathrm{jump}, -, -, *$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$
30: $:=, [\$11], -, \mathrm{temp}$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$
30: $:=, [\$11], -, \mathrm{temp}$
31: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$12$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$
30: $:=, [\$11], -, \mathrm{temp}$
31: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$12$
32: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$13$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$
30: $:=, [\$11], -, \mathrm{temp}$
31: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$12$
32: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$13$
33: $:=, [\$13], -, [\$12]$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=$, i, j, 29
28: jump, $-$, $-$, $*$
29: array, a, i, \$11
30: $:=$, [\$11], $-$, temp
31: array, a, i, \$12
32: array, a, j, \$13
33: $:=$, [\$13], $-$, [\$12]
34: array, a, j, \$14

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \text{i}, \text{j}, 29$
28: $\text{jump}, -, -, *$
29: $\text{array}, \text{a}, \text{i}, \$11$
30: $:=, [\$11], -, \text{temp}$
31: $\text{array}, \text{a}, \text{i}, \$12$
32: $\text{array}, \text{a}, \text{j}, \$13$
33: $:=, [\$13], -, [\$12]$
34: $\text{array}, \text{a}, \text{j}, \$14$
35: $:=, \text{temp}, -, [\$14]$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \text{i}, \text{j}, 29$
28: $\text{jump}, -, -, *$
29: $\text{array}, \text{a}, \text{i}, \$11$
30: $:=, [\$11], -, \text{temp}$
31: $\text{array}, \text{a}, \text{i}, \$12$
32: $\text{array}, \text{a}, \text{j}, \$13$
33: $:=, [\$13], -, [\$12]$
34: $\text{array}, \text{a}, \text{j}, \$14$
35: $:=, \text{temp}, -, [\$14]$
36: $+, \text{i}, 1, \$15$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
29: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$11$
30: $:=, [\$11], -, \mathrm{temp}$
31: $\mathrm{array}, \mathrm{a}, \mathrm{i}, \$12$
32: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$13$
33: $:=, [\$13], -, [\$12]$
34: $\mathrm{array}, \mathrm{a}, \mathrm{j}, \$14$
35: $:=, \mathrm{temp}, -, [\$14]$
36: $+, \mathrm{i}, 1, \$15$
37: $:=, \$15, -, \mathrm{i}$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \text{i}, \text{j}, 29$
28: $\text{jump}, -, -, *$
29: $\text{array}, \text{a}, \text{i}, \$11$
30: $:=, [\$11], -, \text{temp}$
31: $\text{array}, \text{a}, \text{i}, \$12$
32: $\text{array}, \text{a}, \text{j}, \$13$
33: $:=, [\$13], -, [\$12]$
34: $\text{array}, \text{a}, \text{j}, \$14$
35: $:=, \text{temp}, -, [\$14]$
36: $+, \text{i}, 1, \$15$
37: $:=, \$15, -, \text{i}$
38: $-, \text{j}, 1, \$16$

Παράδειγμα (iv)

```
if i <= j then begin
  temp := a[i]; a[i] := a[j]; a[j] := temp;
  i := i+1; j := j-1
end
```

27: $<=, \text{i}, \text{j}, 29$
28: $\text{jump}, -, -, *$
29: $\text{array}, \text{a}, \text{i}, \$11$
30: $:=, [\$11], -, \text{temp}$
31: $\text{array}, \text{a}, \text{i}, \$12$
32: $\text{array}, \text{a}, \text{j}, \$13$
33: $:=, [\$13], -, [\$12]$
34: $\text{array}, \text{a}, \text{j}, \$14$
35: $:=, \text{temp}, -, [\$14]$
36: $+, \text{i}, 1, \$15$
37: $:=, \$15, -, \text{i}$
38: $-, \text{j}, 1, \$16$
39: $:=, \$16, -, \text{j}$

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$

8: $\mathrm{jump}, -, -, *$

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
...
27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
...
27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, *$
...
40: $\mathrm{jump}, -, -, 7$

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, *$
...
27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, 7$
...
40: $\mathrm{jump}, -, -, 7$

Παράδειγμα (v)

```
while i <= j do begin
  while a[i] < a[(m+n) div 2] do i := i+1;
  while a[j] > a[(m+n) div 2] do j := j-1;
  if i <= j then begin
    temp := a[i]; a[i] := a[j]; a[j] := temp;
    i := i+1; j := j-1
  end
end;
```

7: $<=, \mathrm{i}, \mathrm{j}, 9$
8: $\mathrm{jump}, -, -, 41$
...
27: $<=, \mathrm{i}, \mathrm{j}, 29$
28: $\mathrm{jump}, -, -, 7$
...
40: $\mathrm{jump}, -, -, 7$

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort
45: par, a, R, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort
45: par, a, R, –
46: par, i, V, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort
45: par, a, R, –
46: par, i, V, –
47: par, n, V, –

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort
45: par, a, R, –
46: par, i, V, –
47: par, n, V, –
48: call, –, –, quicksort

Παράδειγμα (vi)

```
  quicksort(a, m, j); quicksort(a, i, n)
end;
```

41: par, a, R, –
42: par, m, V, –
43: par, j, V, –
44: call, –, –, quicksort
45: par, a, R, –
46: par, i, V, –
47: par, n, V, –
48: call, –, –, quicksort
49: endu, quicksort, –, –

Τελικός υπολογιστής (i)

- Χαρακτηριστικά:
  - Επεξεργαστής: Intel 8086
  - Λειτουργικό σύστημα: MS-DOS
  - Μοντέλο μνήμης: COM / tiny
    - $\Rightarrow$ Συνολική μνήμη $\leq 64\,\mathrm{K}$
    - $\Rightarrow$ Οργάνωση σε ένα segment
    - $\Rightarrow$ 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
    $\Rightarrow$ 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)
  - Ειδικοί καταχωρητές: ip (instruction pointer) και καταχωρητής σημαιών (flags)

Τελικός υπολογιστής (iii)

- Διευθύνσεις:

$$address = segment * 16 + offset$$

- Μορφή εντολής:

$[\ label\ ]\ \ opname\ \ [\ operand_1\ [\ ,\ operand_2\ ]\ ]$

Τελικός υπολογιστής (iv)

- Εντολές:
  - Μεταφοράς: `mov`, `lea`
  - Αριθμητικών πράξεων: `add`, `sub`, `neg`, `imul`, `idiv`, `cmp`, `cwd`
  - Λογικών πράξεων: `and`, `or`, `xor`, `not`, `test`
  - Άλματος: `jmp`, `jz`, `jnz`, `jl`, `jle`, `jg`, `jge`
  - Διαχείρισης στοίβας: `push`, `pop`
  - Υποπρογραμμάτων: `call`, `ret`
  - Πράξεων κινητής υποδιαστολής (x87 FPU)

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 — $(\texttt{dx}, \texttt{ax}) := \texttt{ax} * op$
- `idiv` op — $\texttt{ax} := (\texttt{dx}, \texttt{ax})$ **div** op; $\texttt{dx} := (\texttt{dx}, \texttt{ax})$ **mod** op
- `cwd` — επέκταση προσήμου του ax στον dx
- `cmp` op_1, op_2 — σύγκρινε τα op_1 και op_2, ενημέρωσε τις σημαίες

Λογικές πράξεις

- `and` op_1, op_2 — $op_1 := op_1$ **and** op_2
- `or` op_1, op_2 — $op_1 := op_1$ **or** op_2
- `not` op — $op :=$ **not** op
- `xor` op_1, op_2 — $op_1 := op_1$ **xor** op_2
- `test` op_1, op_2 — op_1 **and** op_2
  ενημέρωσε τις σημαίες

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
  $\mathrm{sp} := \mathrm{sp} - 2, \quad [\mathrm{sp}] := op$
- pop *op* — αφαίρεση από τη στοίβα
  $op := [\mathrm{sp}], \quad \mathrm{sp} := \mathrm{sp} + 2$

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

Εντολές υποπρογραμμάτων

- `call` *address* — κλήση

$$\texttt{sp} := \texttt{sp} - 2, \quad [\texttt{sp}] := \texttt{ip}, \quad \texttt{ip} := \mathit{address}$$

- `ret` — επιστροφή

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

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

Εντολές x87 FPU (i)

⇒ Ειδική στοίβα 8 καταχωρητών: ST(0), ... ST(7)

- `fld` *source* (load real & push)

```
fld tbyte ptr @real1
```

- `fild` *source* (load integer & push)

```
fild word ptr [bp - 2]
```

- `fstp` *destination* (pop & store real)

```
fld tbyte ptr [bp - 10]
```

- `fistp` *destination* (pop & store integer)

```
fild word ptr [bp - 2]
```

Εντολές x87 FPU (ii)

- `faddp ST(1),ST(0)` $\quad \texttt{ST}(1) := \texttt{ST}(1) + \texttt{ST}(0)$ & pop
- `fsubp ST(1),ST(0)` $\quad \texttt{ST}(1) := \texttt{ST}(1) - \texttt{ST}(0)$ & pop
- `fmulp ST(1),ST(0)` $\quad \texttt{ST}(1) := \texttt{ST}(1) * \texttt{ST}(0)$ & pop
- `fdivp ST(1),ST(0)` $\quad \texttt{ST}(1) := \texttt{ST}(1)/\texttt{ST}(0)$ & pop
- `fchs` $\quad \texttt{ST}(0) := -\texttt{ST}(0)$
- `fcompp` $\quad \texttt{ST}(1) \lessgtr \texttt{ST}(0)$ & pop both
- `fstsw` *destination* $\quad$ (store x87 FPU flags)

```
fstsw  ax
fstsw  word ptr [bp - 2]
```

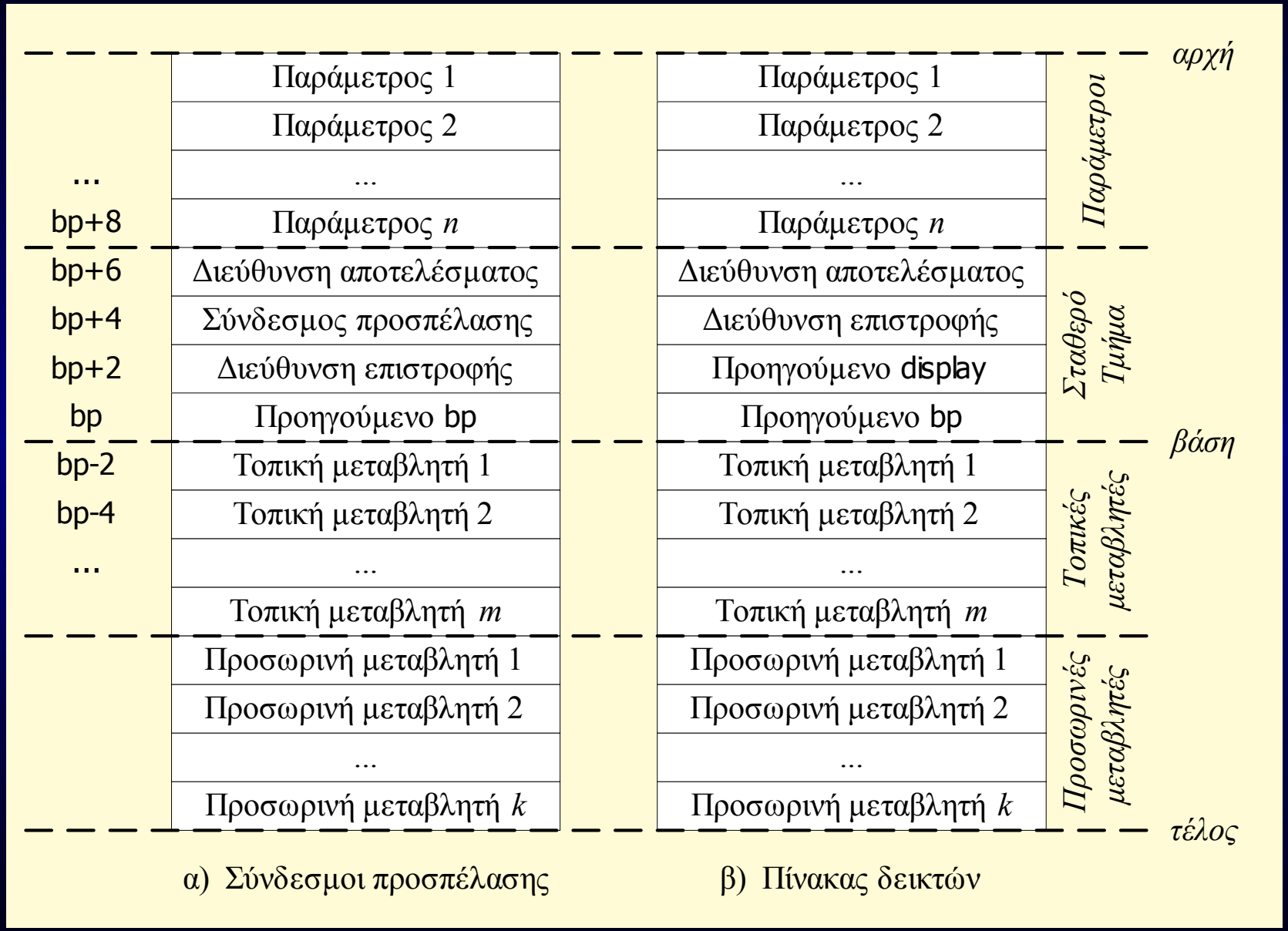
Διαχείριση μνήμης (i)

- Δομή ενοτήτων (block structure)
  - Μη τοπικά δεδομένα

Διαχείριση μνήμης (i)

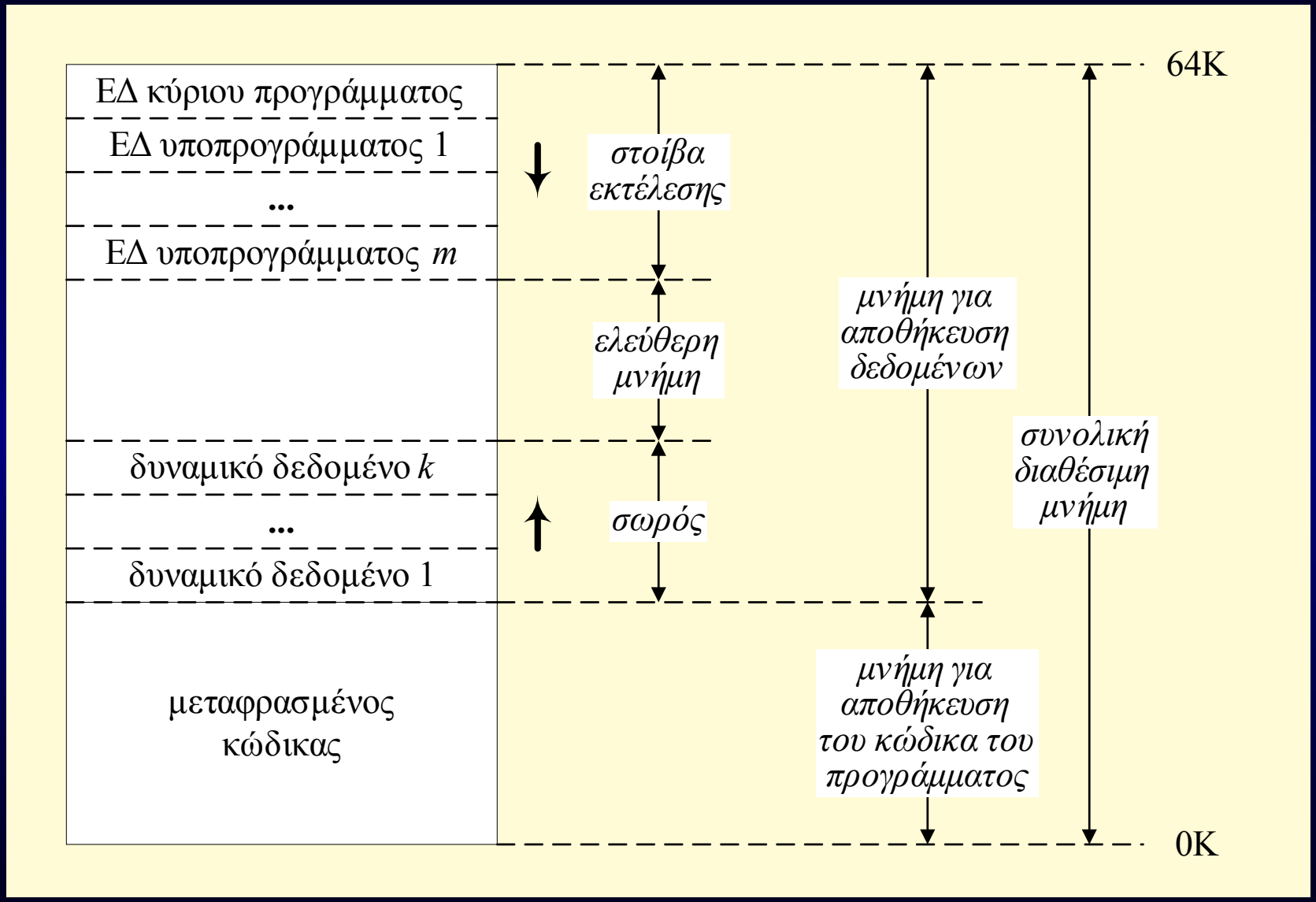
- Δομή ενοτήτων (block structure)
  - Μη τοπικά δεδομένα
- Εγγράφημα δραστηριοποίησης (activation record / frame)
  - Παράμετροι
  - Αποτέλεσμα
  - Πληροφορίες κατάστασης μηχανής
  - Τοπικές μεταβλητές
  - Προσωρινές μεταβλητές

Διαχείριση μνήμης (ii)



α) Σύνδεσμοι προσπέλασης β) Πίνακας δεικτών

Διαχείριση μνήμης (iii)



Προσπέλαση ονομάτων

- Τοπικά: $[\texttt{bp} + \mathit{offset}]$
- Μη τοπικά: $[\texttt{si} + \mathit{offset}]$
  - $\Rightarrow$ 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 ΕΔ της p τοποθετείται ένα σύνδεσμος προς τη βάση του ΕΔ 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