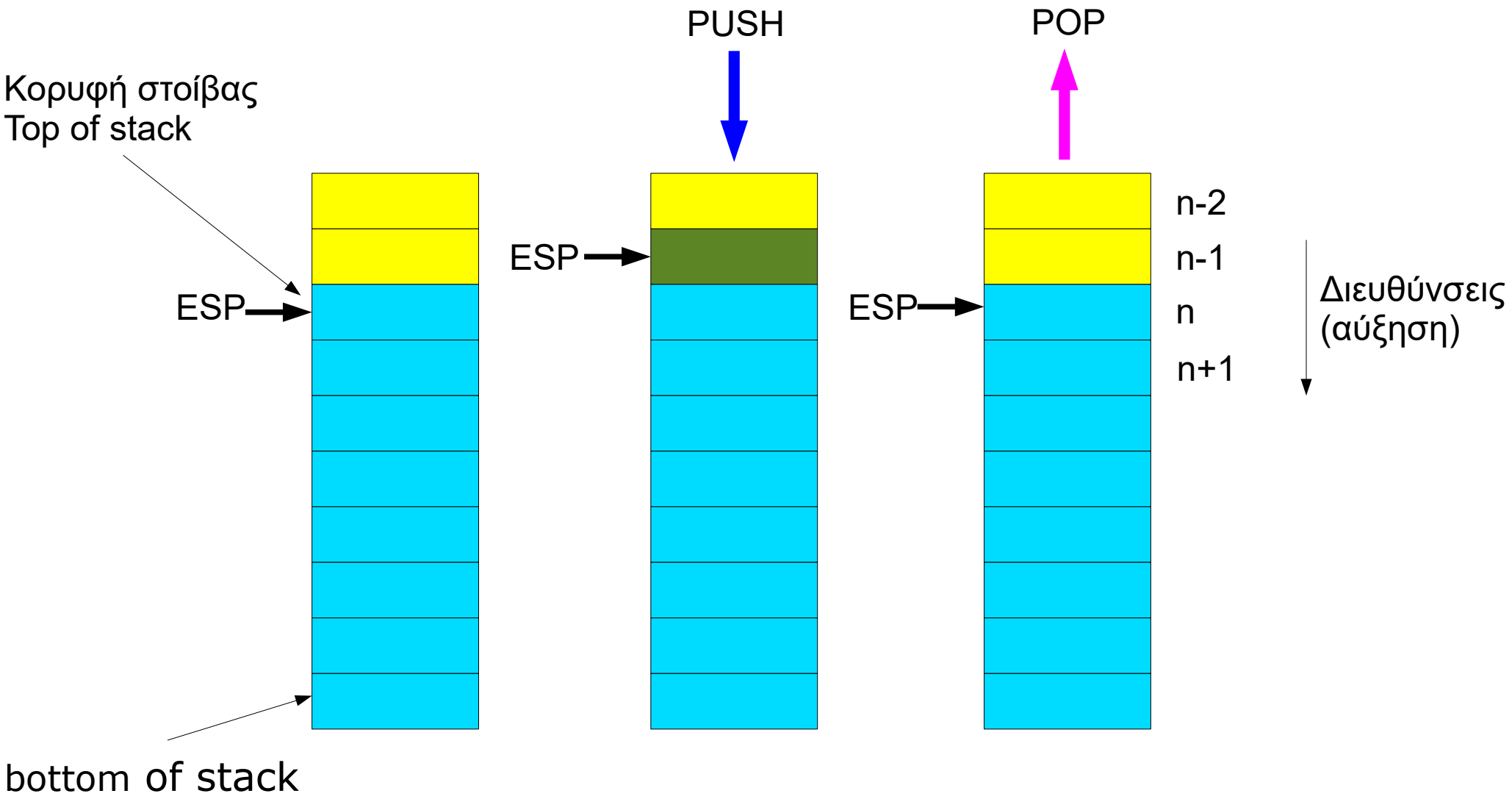
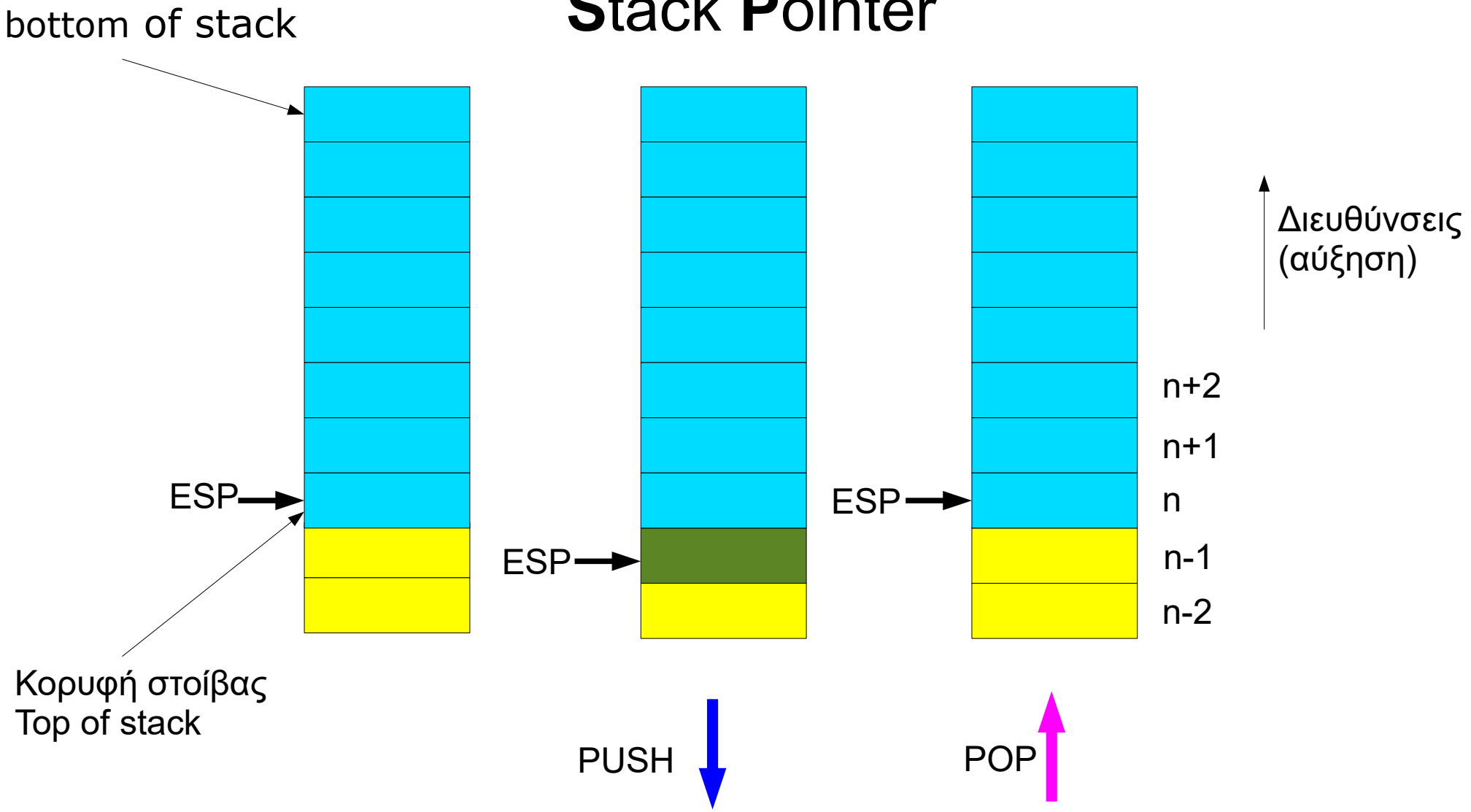
PROCEDURE CALLS, INTERRUPTS, AND EXCEPTIONS

Στοίβα Stack
Stack **P**ointer



Last-In-First-Out (LIFO)

Στοίβα Stack
Stack **P**ointer



Αποθήκευση στη στοίβα

Ανάκτηση από τη στοίβα

```
PUSH EAX
PUSH EBX
PUSH ECX
PUSH EDX
```

```
POP EDX
POP ECX
POP EBX
POP EAX
```

```
POP EAX
POP ECX
POP EBX
POP EDX
```

```
POP EDX
POP ECX
POP EAX
POP EAX
```

EAX ⟷ EDX

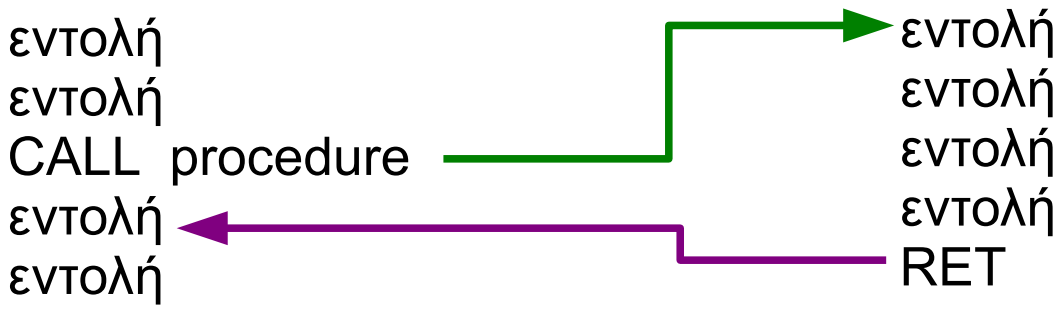
```
PUSH EDX
POP EIP
```

Ιδιότητες

- Η στοίβα αυξάνεται προ τα κάτω - προς μικρότερες διευθύνσεις (the stack grows down in memory)
- Η εντολή PUSH μετακινεί τον δείκτη ESP σε μικρότερη διεύθυνση
- Η εντολή POP μετακινεί το δείκτη ESP σε μεγαλύτερη διεύθυνση
- Η αρχική τιμή του δείκτη ESP, όταν η στοίβα είναι κενή, είναι το κάτω μέρος (πυθμένας) της στοίβας (bottom of stack)
- Η τρέχουσα θέση του δείκτη ESP είναι η κορυφή της στοίβας (top of stack) – η θέση στην οποία προσθέτουμε ή αφαιρούμε στοιχεία
- Αν η στοίβα έχει ορισμένο μέγεθος και προσθέσουμε περισσότερα στοιχεία θα γίνει υπερχείλιση (stack overflow)

Ρυθμίσεις στοίβας

- Επιλογή ενός τμήματος μνήμης για τη στοίβα (Stack Segment)
- Τοποθέτηση τού επιλογέα (segment selector) στον καταχωρητή SS
- Φόρτωση του ESP με την διεύθυνση του πυθμένα της στοίβας
- Η τιμή αυτή πρέπει να ευθυγραμμίζεται με το όριο λέξης 16 ή 32 bit αν ή στοίβα είναι οργανωμένη σε 16 ή 32 bit.(Πρέπει να είναι άρτιος αριθμός, n*2 ή n*4 bytes)
- Η οργάνωση της στοίβας προσδιορίζεται από το D-flag.
- Οι segment registers αν και 16bits αποθηκεύονται στη 32bit stack σε 4 bytes



- CALL αποθηκεύει τη διεύθυνση της επόμενης εντολής (return address) στη stack και αρχίζει να εκτελεί τις εντολές της procedure.
- Η procedure μπορεί να αλλάξει τη τιμή του ESP. Ο προγραμματιστής πρέπει να επαναφέρει τη τιμή του ESP πριν από την εντολή RET.
- RET επιστρέφει στην εκτέλεση των εντολών μετά την CALL.

Near CALL

Ο κώδικας της procedure είναι στο ίδιο code segment.

- **CALL** : Ο κώδικας της procedure είναι στο ίδιο code segment.
  - EIP → stack
  - EIP ← διεύθυνση πρώτης εντολής της procedure
  - Εκτέλεση εντολών της procedure
- **RET** :
  - EIP ← Κορυφή της stack
  - Εκτέλεση των εντολών μετά την CALL

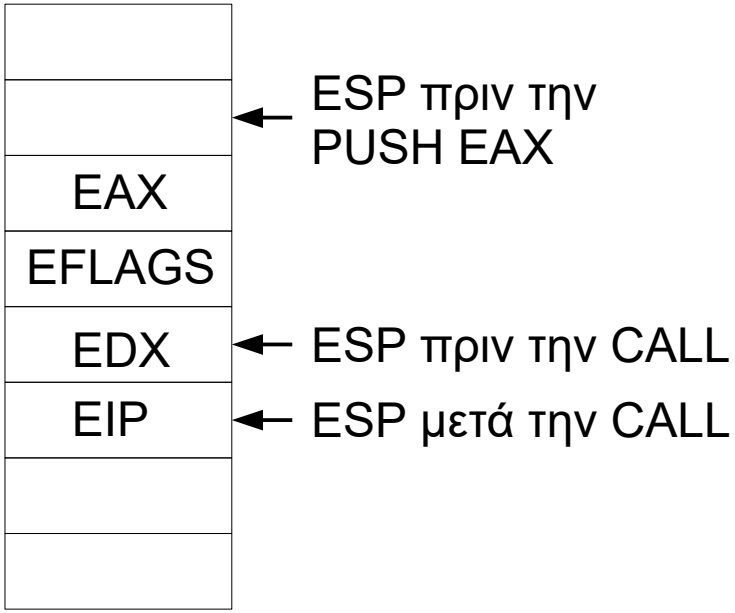
Far CALL

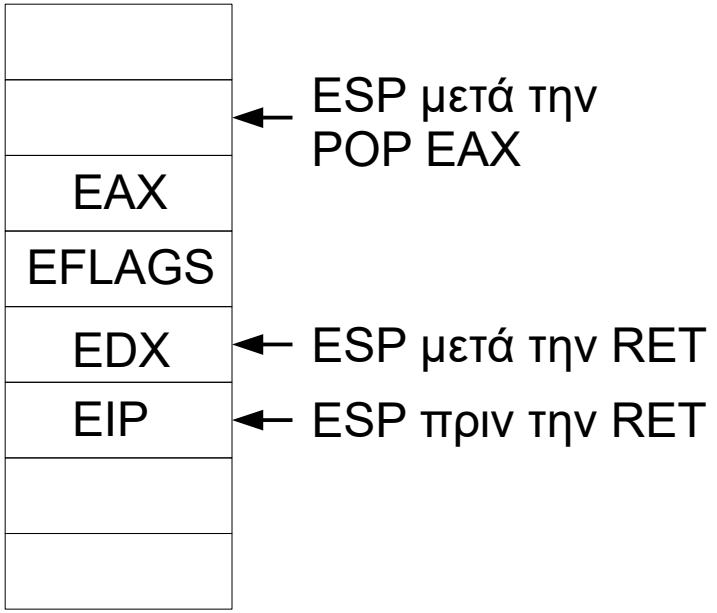
Ο κώδικας της procedure είναι σε άλλο code segment.

- **CALL** :
  - CS → stack
  - EIP → stack
  - CS ← Code segment procedure
  - EIP ← διεύθυνση πρώτης εντολής της procedure (offset)
  - Εκτέλεση εντολών της procedure
- **RET** :
  - EIP ← Κορυφή της stack
  - CS ← Κορυφή της stack
  - Εκτέλεση των εντολών μετά την CALL

Η procedure αλλάζει τους καταχωρητές

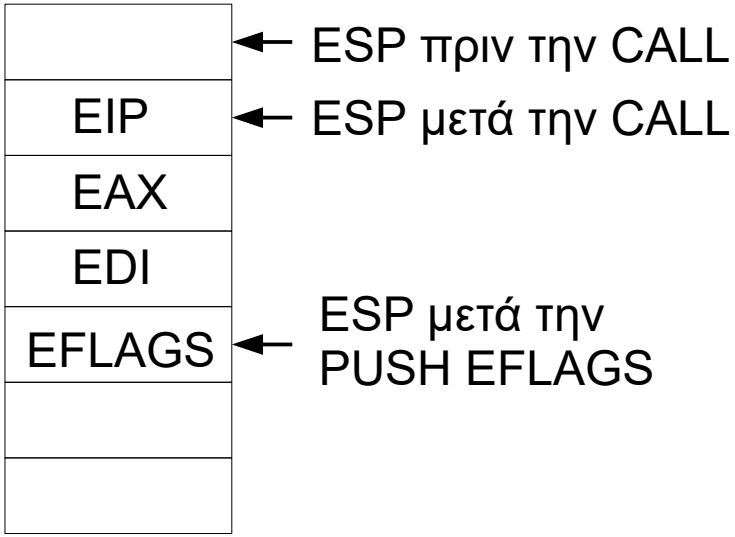
```
PUSH EAX
PUSH EFLAGS
PUSH EDX
CALL near_prodedure
POP EDX
POP EFLAGS
POP EAX
```

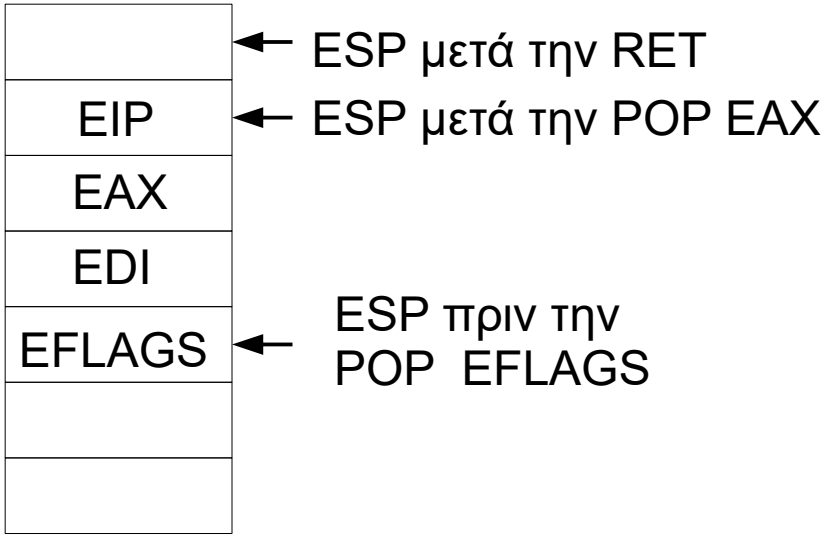




Η procedure αλλάζει τους καταχωρητές

```
near_prodedure:
PUSH EAX
PUSH EDI
PUSH EFLAGS
.....
POP EFLAGS
POP EDI
POP EAX
RET
```





Ο ESP δεν δείχνει στη διεύθυνση επιστροφής

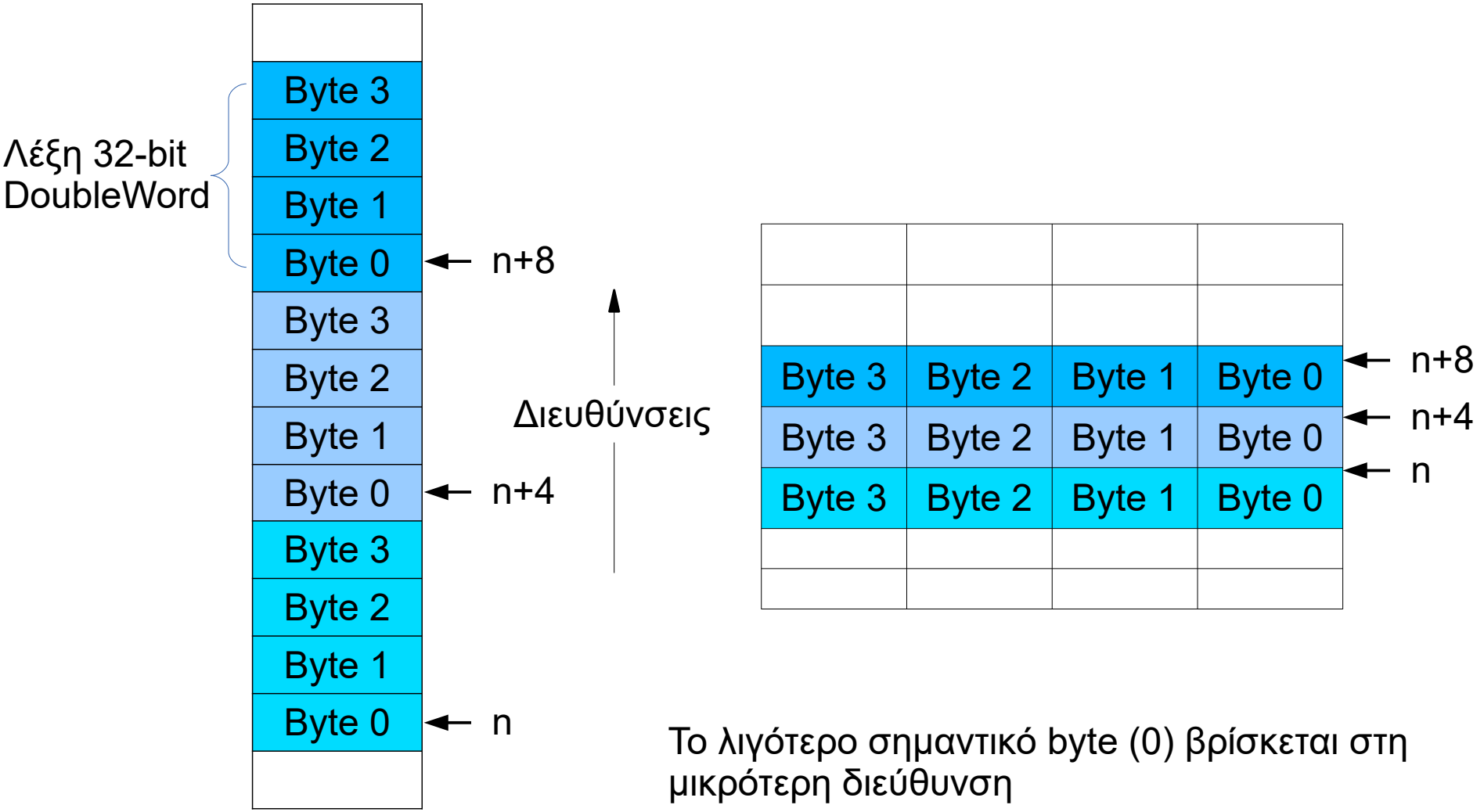
Κλήση με πέρασμα παραμέτρων

Κλήση συνάρτησης σε C

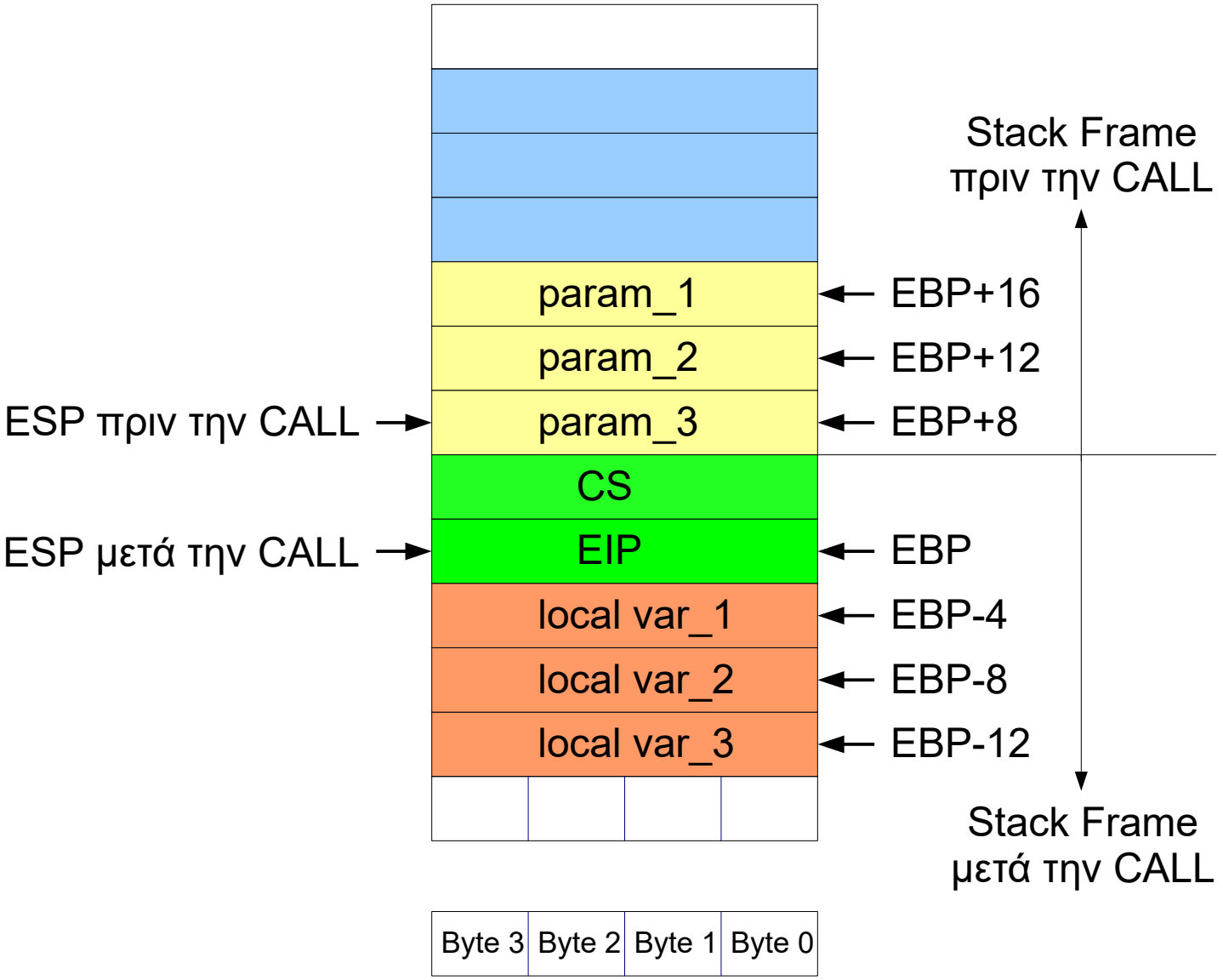
```
int param_1=10;
int param_2=20;
int param_3=30;
function_1(param_1 , param_2 , param_3)
```

Κλήση διαδικασίας σε Assembly

```
PUSH [param_1]
PUSH [param_2]
PUSH [param_3]
CALL _function_1
```


Λέξη 32-bit
DoubleWord
Byte 3
Byte 2
Byte 1
Byte 0
n+8
Byte 3
Byte 2
Byte 1
Byte 0
n+4
Byte 3
Byte 2
Byte 1
Byte 0
n
Διευθύνσεις
Byte 3 Byte 2 Byte 1 Byte 0
n+8
Byte 3 Byte 2 Byte 1 Byte 0
n+4
Byte 3 Byte 2 Byte 1 Byte 0
n
Το λιγότερο σημαντικό byte (0) βρίσκεται στη μικρότερη διεύθυνση

Stack frame



EBP : δείκτης στο SS

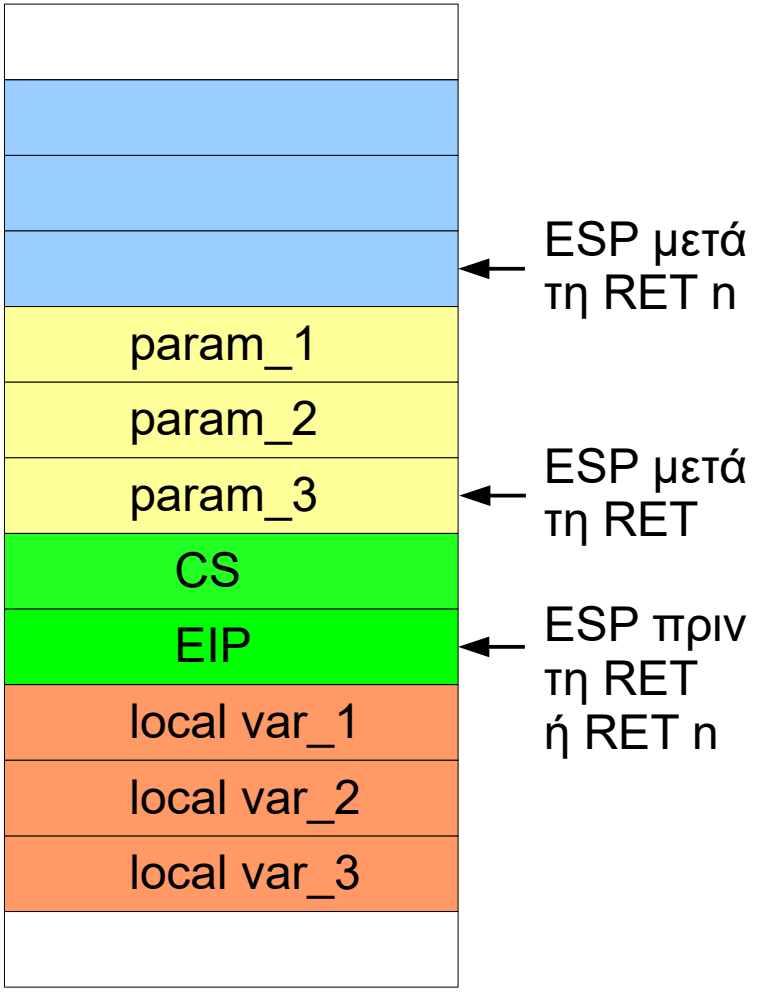
Στην αρχή της procedure
EBP = ESP

param_3 : [EBP+8]
param_2 : [EBP+12]
param_1 : [EBP+16]

local_var_1 : [EBP-4]
local_var_1 : [EBP-8]
local_var_3 : [EBP-12]

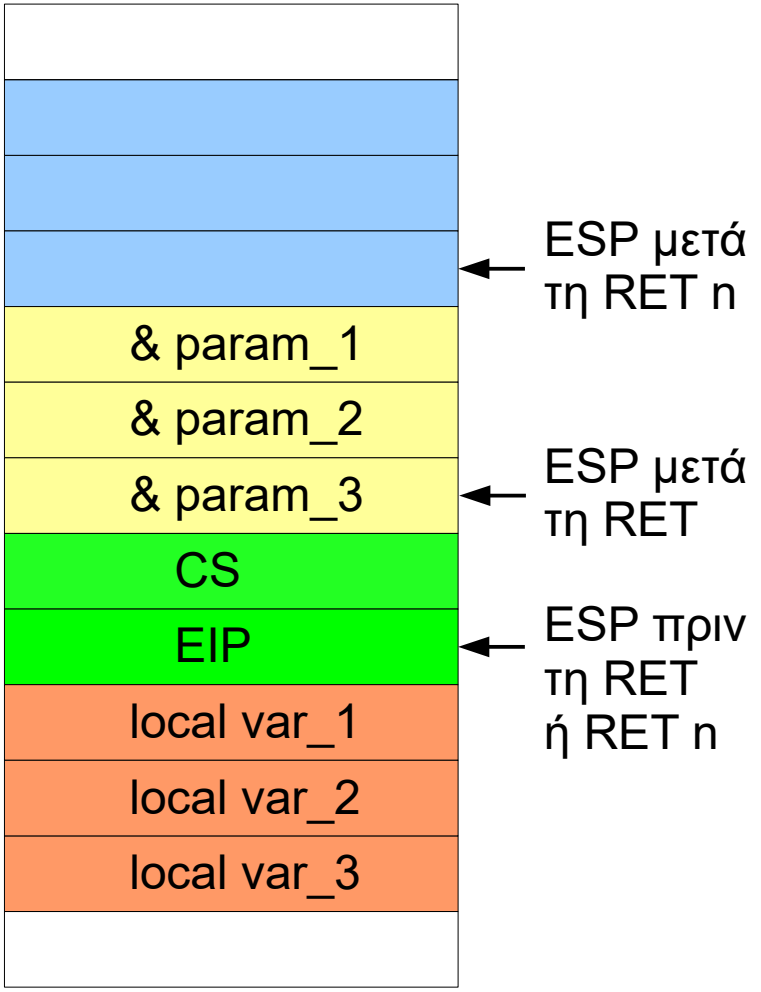
```
void function_1(int x, int y, int z)
{
  int a, int b, int c; // local
  ...
  return;
}
```

```
_function_1:
  PUSH EBP ;  η  συνάρτηση αλλάζει τον EBP
  MOV EBP, ESP ; EBP δείχνει στη διεύθυνση επιστροφής
  ;x = [EBP + 8] , y = [EBP + 12] , z = [EBP + 16]
  ;a = [EBP - 4] , b = [EBP - 8] , c = [EBP – 12]
  .....
  .....
  .....
  MOV ESP, EBP ; αν ο ESP άλλαξε
  POP EBP ; επαναφορά της τιμής του EBP πριν την κλήση
  RET 12 ; αποδέσμευση της μνήμης που καταλαμβάνουν
           τα x,y,z αν δεν χρειάζονται
```



```
void function_1(int &x, int &y, int &z)
{
  int a, int b, int c; // local
  ...
  return;
}
```

```
_function_1:
  PUSH EBP ;  η  συνάρτηση αλλάζει τον EBP
  MOV EBP, ESP ; EBP δείχνει στη διεύθυνση επιστροφής
  ;x = *[EBP + 8] , y = *[EBP + 12] , z = *[EBP + 16]
  ;a = [EBP - 4] , b = [EBP - 8] , c = [EBP – 12]
  ..............
  ..............
  ..............
  MOV ESP, EBP ; αν ο ESP άλλαξε
  POP EBP ; επαναφορά της τιμής του EBP πριν την κλήση
  RET 12 ; αποδέσμευση της μνήμης που καταλαμβάνουν
             τα x,y,z αν δεν χρειάζονται
```



Στοίβα σε 64bit

- Ο ESP γίνεται 64 bit
- Ο SS δείχνει για αρχή του segment μηδέν
- Οι segment registers αποθηκεύονται σε 8 bytes

Πέρασμα παραμέτρων

- Μέσα σε καταχωρητές γενικής χρήσης, εκτός των ESP,EBP

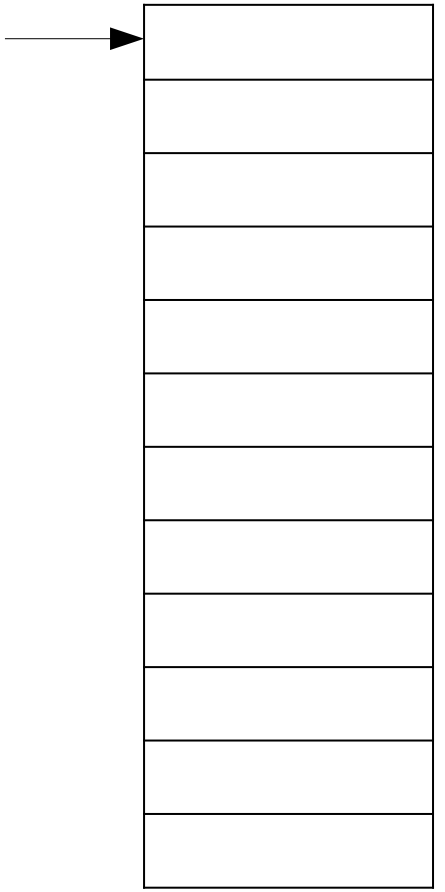
```
MOV EAX,10
MOV EBX,5
MOV ECX,[var_address]
CALL procedure
```

Η procedure μπορεί να επιστρέψει τιμές με τους καταχωρητές

Argument List :

Παράμετροι της procedure
(π.χ ένας πίνακας array)

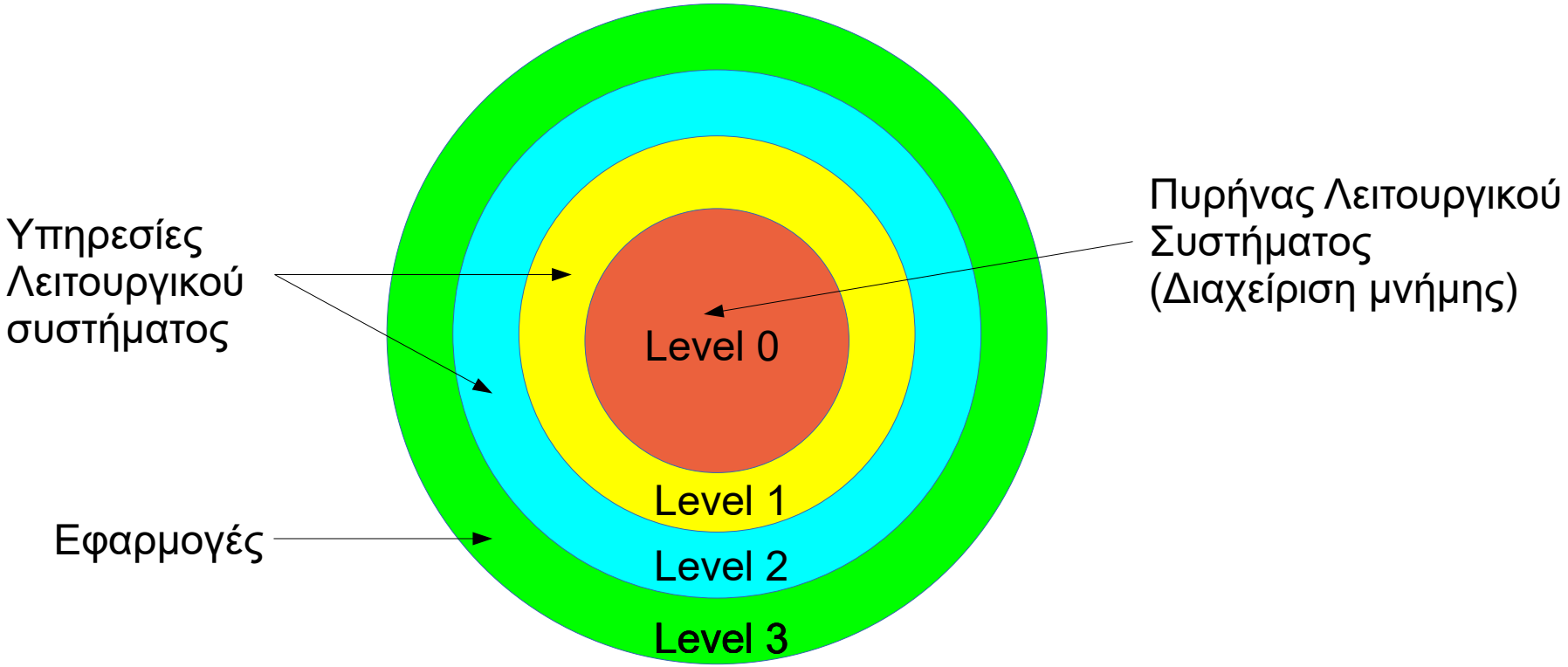
Διεύθυνση πρώτου στοιχείου
(array_address)



```
PUSH array_address
CALL procedure
```

```
MOV  AX , array_address
CALL procedure
```

Privilege Levels



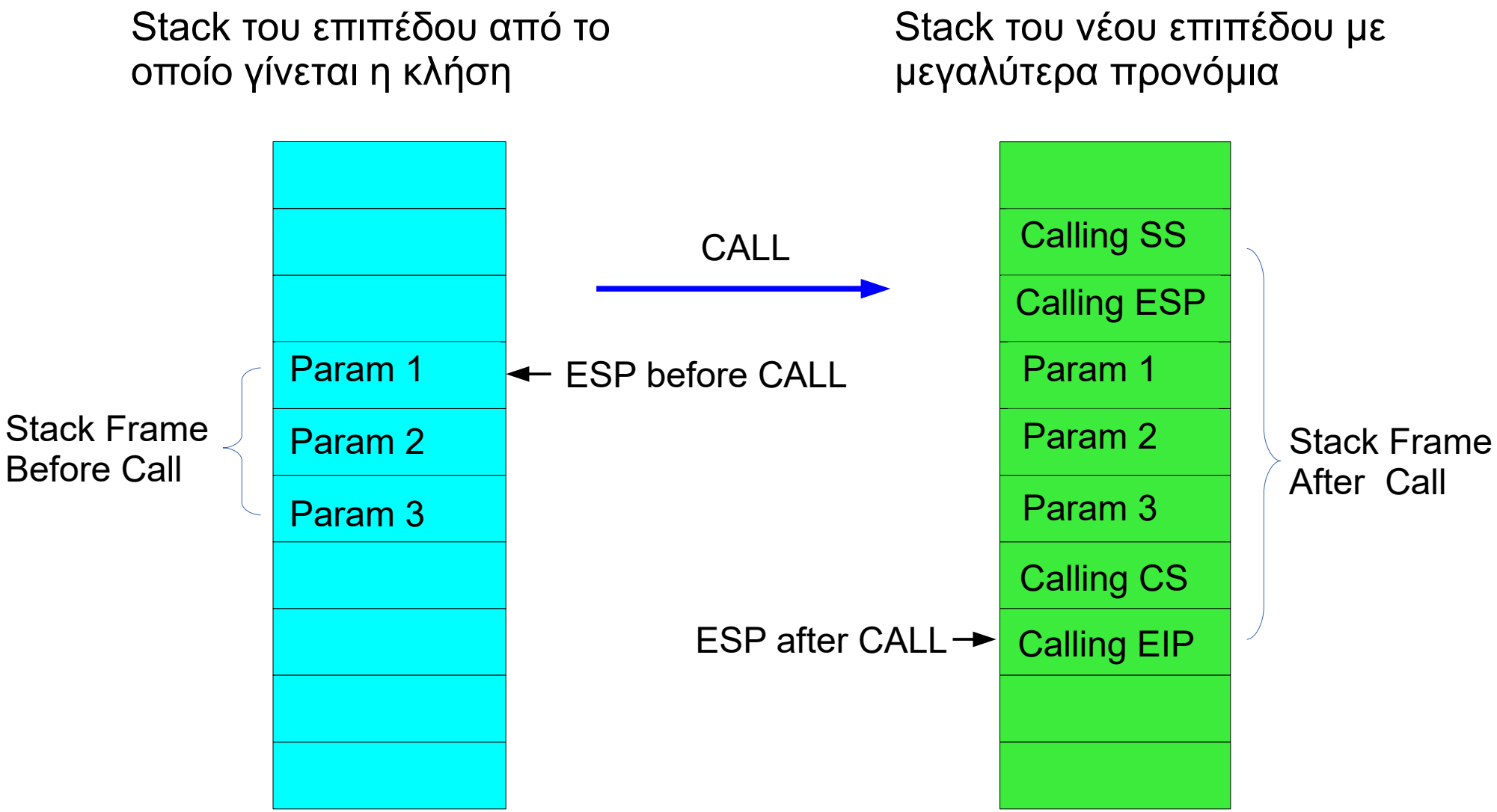
Προνόμια_0 >Προνόμια_1>Προνόμια_2>Προνόμια_3

Κλήση πρός επίπεδο με μεγαλύτερο δείκτη : χωρίς περιορισμούς
Κλήση προς επίπεδο με μικρότερο δείκτη : με περιορισμούς μέσω gate

Κλήση σε επίπεδο με μεγαλύτερα προνόμια Call Gates

- Διαδικασία όμοια με Far Call: Χρησιμοποιείται μια ειδική δομή **call gate descriptor**
  - Δικαιώματα πρόσβασης
  - CS της διαδικασίας που καλείται
  - EIP της διαδικασίας που καλείται
- Κάθε επίπεδο έχει δική του stack.
- Οι τιμές των καταχωρητών SS , ESP για το επίπεδο 3 βρίσκονται στους καταχωρητές.
- Οι τιμές των καταχωρητών SS , ESP για τα επίπεδα 2,1,0 βρίσκονται στο **T**ask **S**tate **S**egment
- Το επίπεδο προνομίων πρόσβασης κάθε προγράμματος ορίζεται από τα 2 bits CPL (Current Privilege Level) που υπάρχουν στις ιδιότητες κάθε code segment.

- Ελεγχος δικαιωμάτων πρόσβασης
- Οι καταχωρητές SS,ESP,CS,EIP αποθηκεύονται προσωρινά σε εσωτερικούς
- Στούς SS,ESP φορτώνονται οι τιμές για τη stack του νέου επιπέδου από το TSS
- Οι προσωρινά αποθηκευμένες τιμές των SS,ESP μεταφέρονται στη νέα stack
- Οι τιμές των παραμέτρων για την διαδικασία που καλείται αποθηκεύονται στη νέα stack
- Οι προσωρινά αποθηκευμένες τιμές των CS,EIP μεταφέρονται στη νέα stack
- Οι καταχωρητές CS,EIP φορτώνονται με τη διεύθυνση της διαδικασίας που καλείται
- Αρχίζει η εκτέλεση της διαδικασίας στο νέο επίπεδο

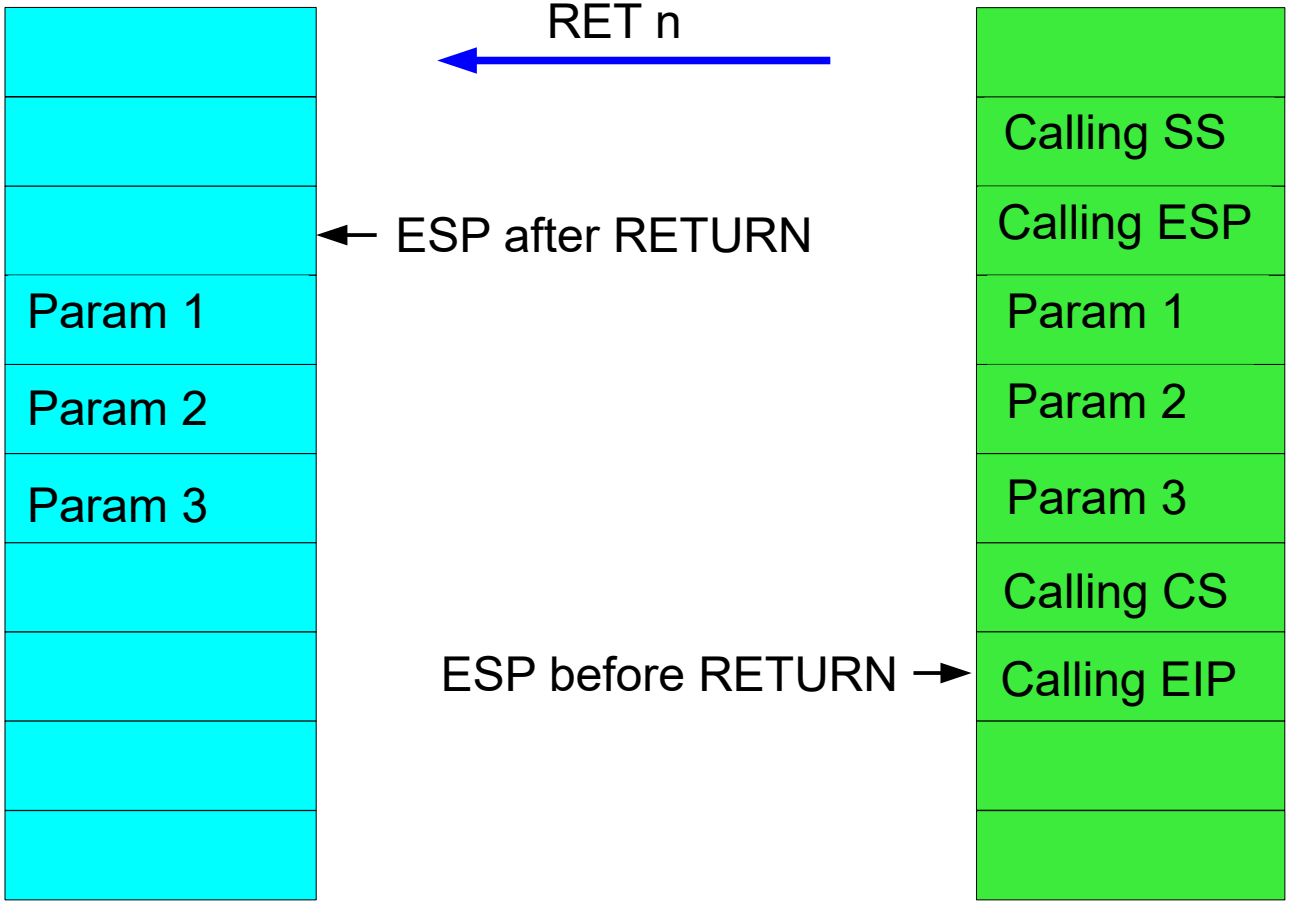

Stack του επιπέδου από το οποίο γίνεται η κλήση
Stack του νέου επιπέδου με μεγαλύτερα προνόμια
CALL
Param 1
ESP before CALL
Param 2
Param 3
Stack Frame Before Call
Calling SS
Calling ESP
Param 1
Param 2
Param 3
Calling CS
Calling EIP
ESP after CALL
Stack Frame After Call

Επιστροφή από επίπεδο με μεγαλύτερα προνόμια

- Ελεγχος δικαιωμάτων πρόσβασης
- Επαναφορά στους CS,EIP των τιμών πριν από τη κλήση
- Αν δίνεται η παράμετρος n (RET n), ο καταχωρητής ESP αυξάνεται ώστε n bytes να αποδεσμευτούν και από τις δύο stack
- Επαναφορά στους SS,ESP των τιμών πριν από τη κλήση.
- Συνεχίζεται η εκτέλεση του προγράμματος (πριν απο τη κλήση).

Stack του επιπέδου από το οποίο γίνεται η κλήση

Stack του νέου επιπέδου με μεγαλύτερα προνόμια



Διακλαδώσεις σε 64-Bit Mode

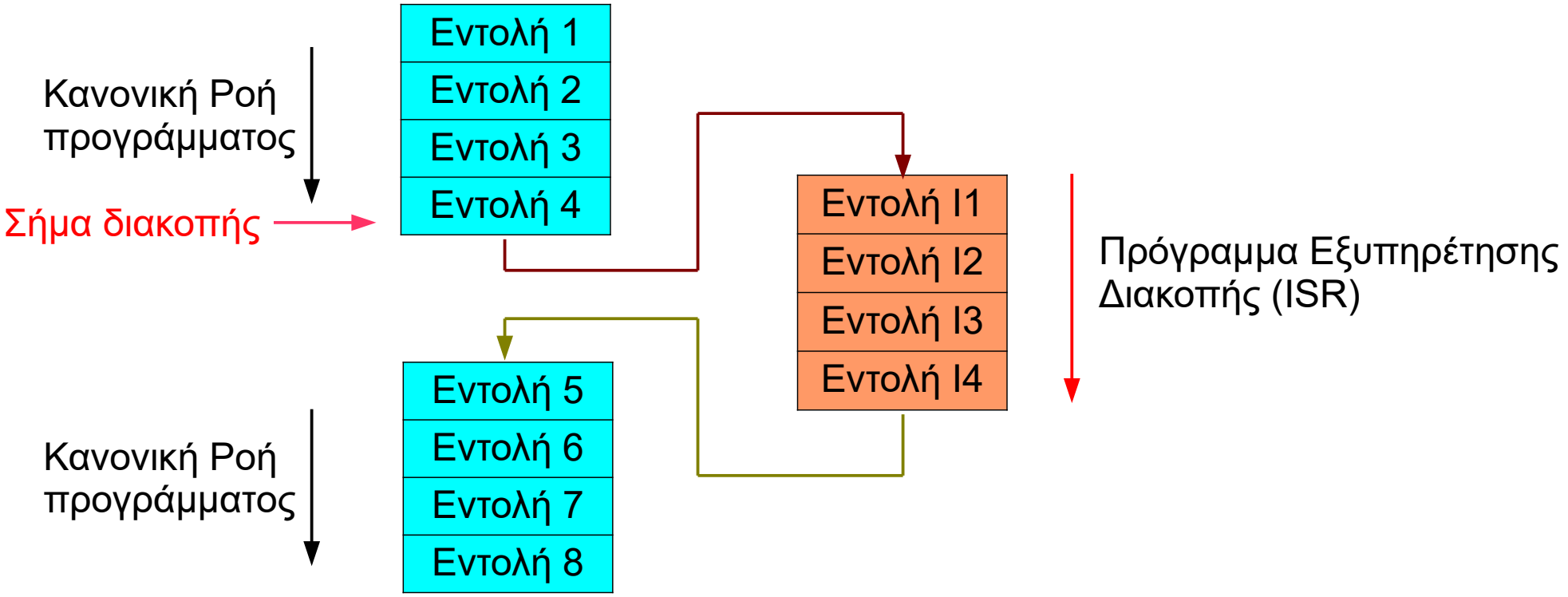
- ΟΙ διευθύνσεις των εντολών διακλάδωσης είναι 64bit (CALL, RET , JNZ , LOOP κ.λ.π.)
- Σε 64-bit mode και compatibility mode υπάρχουν 64-bit call-gate descriptors

Διακοπές (Interrupts)
Εξαιρέσεις (Exceptions)

- Η διακοπή μπορεί είναι ένα ασύγχρονο γεγονός (Hardware Interrupt) ή ένα σύγχρονο γεγονός (Software Interrupt, Exception)
- **Hardware Interrupt**: Προκαλείται από μια περιφερειακή συσκευή (δίσκος) ή μια συσκευή εισόδου-εξόδου (π.χ. Πληκτρολόγιο, ποντίκι κ.α.). Η διαδικασία ενεργοποίησης ονομάζεται Interrupt Request (IRQ)
- **Software Interrupt**: Προκαλείται από μία εντολή INT n
- **Exception**: Προκαλείται από την εκτέλεση μιας εντολής (π.χ. Διαίρεση με μηδέν, λάθος κωδικός εντολής)

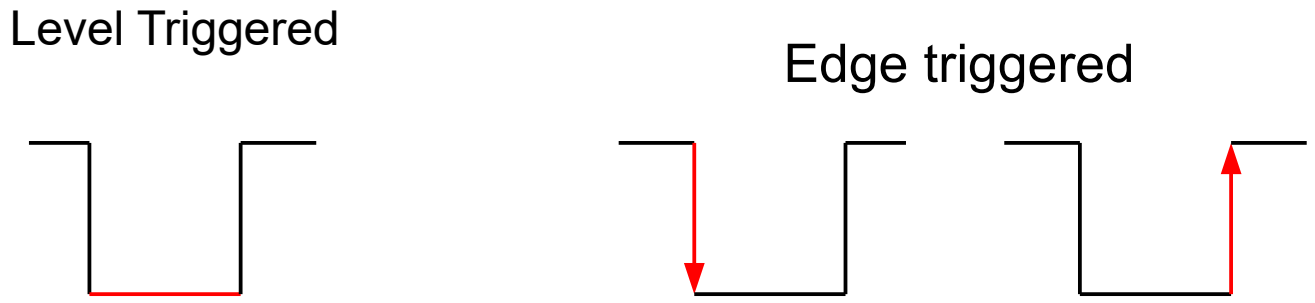
Interrupt **S**ervice **R**outine

Ο επεξεργαστής, αφού εκτελέσει την παρούσα εντολή, διακόπτει την εκτέλεση του προγράμματος, αποθηκεύει την κατάσταση που βρίσκεται και εκτελεί το πρόγραμμα εξυπηρέτησης της διακοπής (Interrupt Handler, **I**nterrupt **S**ervice **R**outine)

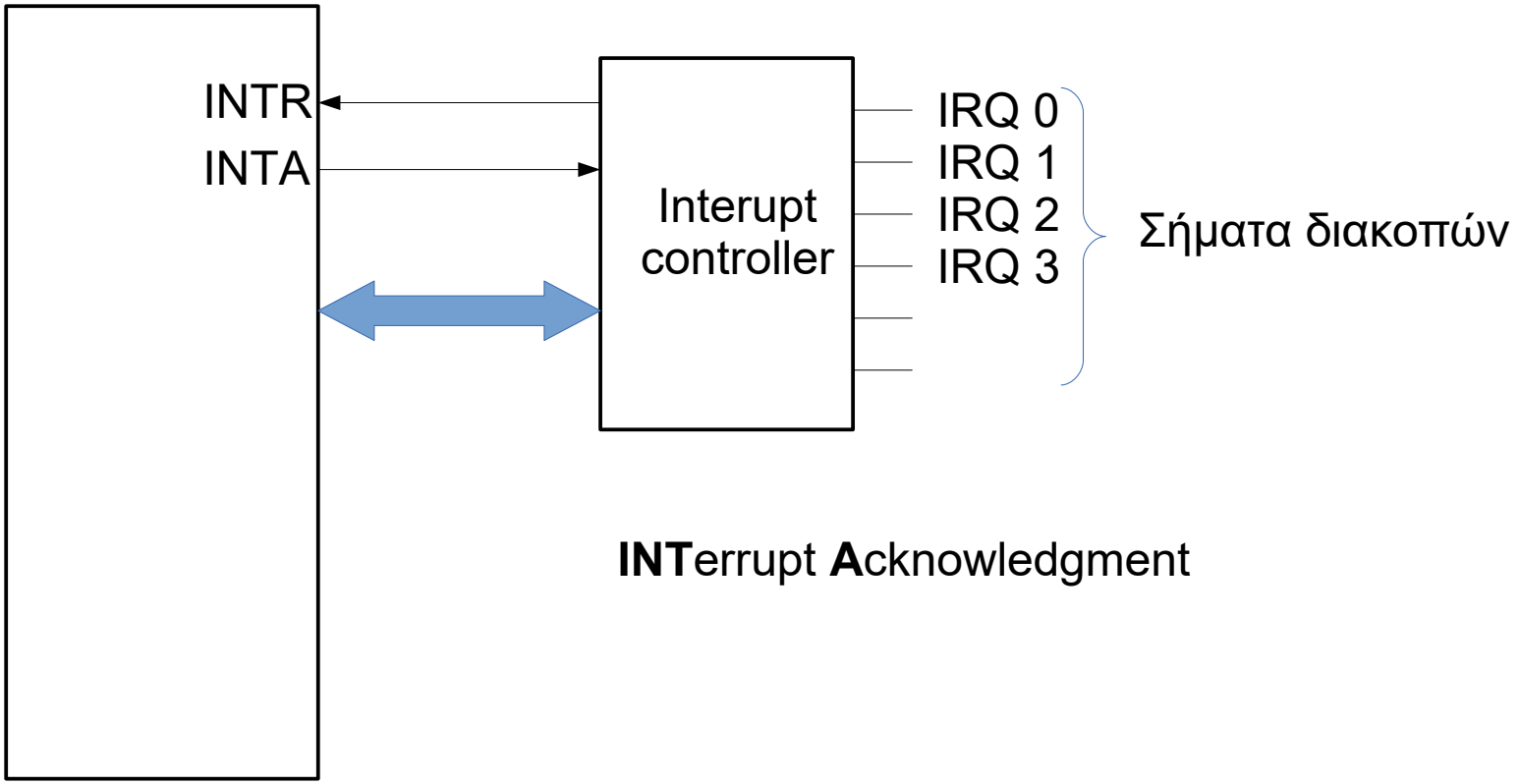


Οταν ολοκληρωθεί το πρόγραμμα εξυπηρέτησης της διακοπής, επιστρέφει στο κυρίως πρόγραμμα.

- Ο επεξεργαστής έχει ένα ή περισσότερους ακροδέκτες για σήματα διακοπών
- Maskable Interrupts: Ενα bit στον καταχωρητή EFLAGS προσδιορίζει αν ο επεξεργαστής θα εξυπηρετεί τις διακοπές ή θα τις αγνοεί (Interrupt Enable Flag, IF 0:clear , 1:enable).
- **N**om **M**askable **I**nterrupt:Η διακοπή εξυπηρετείται ανεξάρτητα από τη τιμή του IF
- Level Edge triggered



Interrupt controller

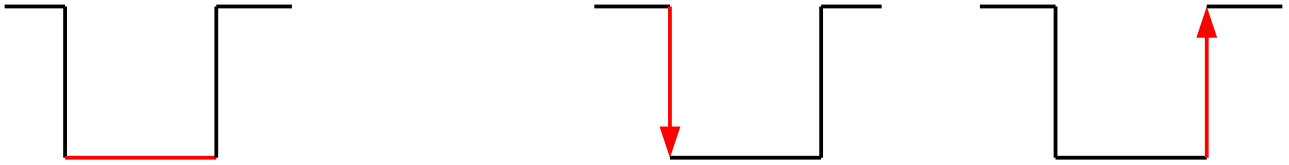


INTerrupt **A**cknowledgment

Ο ελεγκτής διακοπών στέλνει σήμα στον επεξεργαστή και τον τύπο της διακοπής μέσω του data bus

Προγραμματιζόμενες προτεραιότητες

Level / Edge Triggered

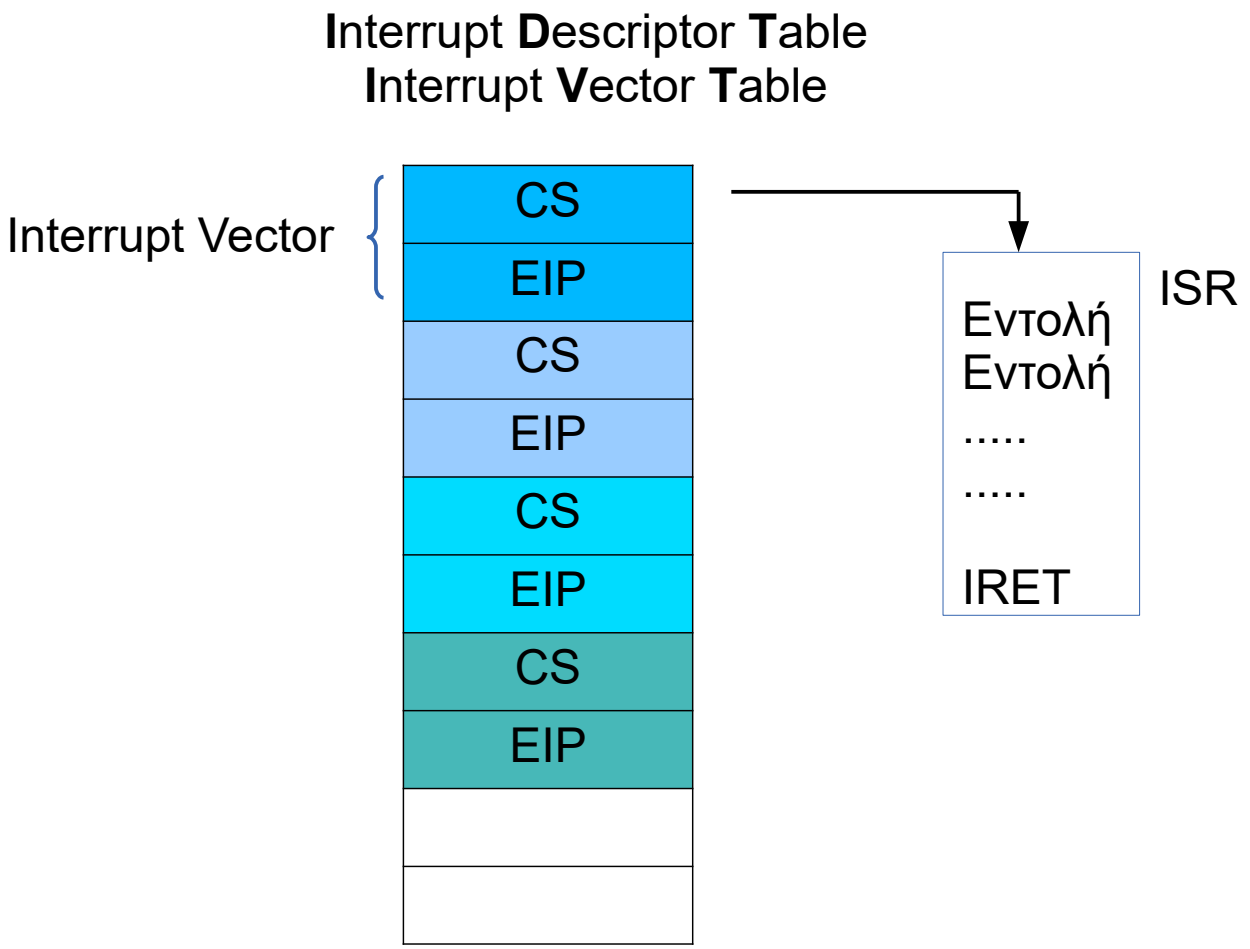


Level triggered

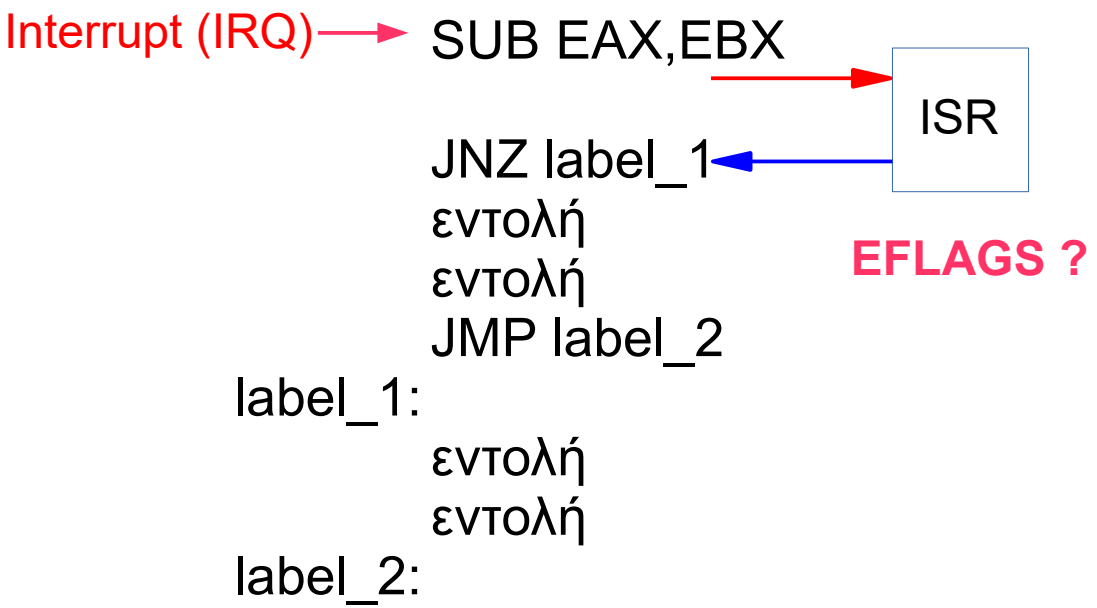
- Εάν η διάρκεια του παλμού είναι πολύ μικρή μπορεί να μην ανιχνευθεί από τον επεξεργαστή.
- Εάν η διάρκεια του παλμού είναι πολύ μεγάλη είναι δυνατόν μετά την εξυπηρέτηση της διακοπής να προκαλέσει νέα διακοπή.
- Κατάλληλη όταν δύο ή περισσότερες διακοπές χρησιμοποιούν την ίδια γραμμή (shared interrupts)

Edge triggered

- Το αίτημα διακοπής ανιχνεύεται πάντοτε.
- Εάν δύο διακοπές χρησιμοποιούν την ίδια γραμμή και οι παλμοί έρθουν ταυτόχρονα μόνα η μια διακοπή θα ανιχνευθεί.

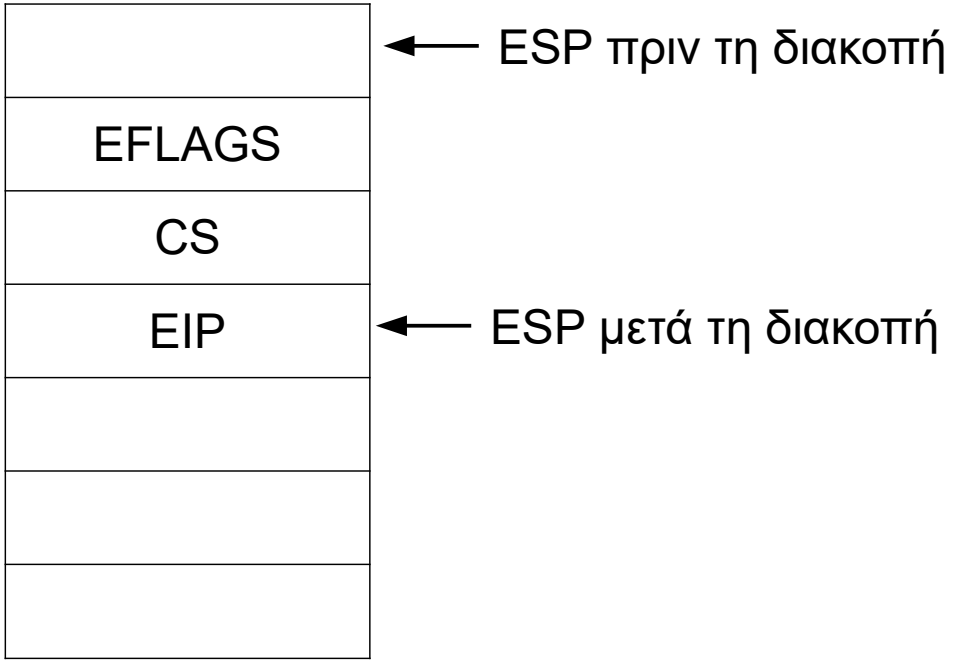

Interrupt Descriptor Table
Interrupt Vector Table
Interrupt Vector
CS
EIP
CS
EIP
CS
EIP
CS
EIP
ISR
Εντολή
Εντολή
.....
.....
IRET

Call - Interrupt



Interrupt gate – trap gate

- Το πρόγραμμα που εκτελείται είναι στο ίδιο επίπεδο με την ρουτίνα διακοπής (ISR) - δεν αλλάζει η stack
  - EFLAGS,CS,EIP ⟶ stack
  - CS,EIP ⟵ interrupt gate
  - Αν είναι interrupt gate IF = 0 : Αγνοεί άλλα σήματα διακοπής. Αν θέλουμε να επιτρέψουμε τις διακοπές πρέπει να γράψουμε στο bit IF '1'
  - Αν είναι trap gate το bit IF δεν αλλάζει τιμή
  - Εκτέλεση ISR



- Ο κώδικας του προγράμματος είναι σε επίπεδο με μικρότερα προνόμια από την ISR – αλλαγή stack
  - SS, ESP, EFLAGS, CS, EIP αποθηκεύονται προσωρινά
  - SS,ESP ← TSS (stack του επιπέδου ISR)
  - Αποθηκευμένες τιμές SS, ESP, EFLAGS, CS, EIP → νέα stack
  - CS, EIP ← interrupt gate
  - IF = 0 αν είναι interrupt gate
  - Εκτέλεση ISR στο νέο επίπεδο

IRET

- Χωρίς αλλαγή stack
  - CS,EIP ← stack
  - EFLAGS← stack
- Με αλλαγή stack
  - Ελεγχος δικαιωμάτων πρόσβασης
  - EFLAGS← stack
  - CS,EIP ← stack
  - SS,ESP ← stack (επιστροφή στη stack του αρχικού επιπέδου)

Εξαιρέσεις Exceptions

Fault

- Η εντολή που αποκωδικοποιήθηκε δεν μπορεί να εκτελεστεί
- Η κατάσταση ανιχνεύεται **πριν** αυξηθεί ο EIP
- Ο επεξεργαστής αποθηκεύει στη stack τις απαραίτητες πληροφορίες
- Καλείται η κατάλληλη διαδικασία (ρουτίνα fault handler)
- Αν το πρόβλημα μπορεί να αποκατασταθεί συνεχίζεται η εκτέλεση του προγράμματος
- Παραδείγματα:
  - Η εντολή προσπαθεί να γράψει σε περιοχή μνήμης “μόνο για ανάγνωση”
  - Πρόσβαση σε μία σελίδα που δεν υπάρχει στη μνήμη
  - Εντολή που απαιτεί αυξημένα προνόμια
  - Διαίρεση με μηδέν

Trap

- Ο έλεγχος περνά σε μια διαδικασία debugger μετά τη εκτέλεση μιας εντολής
- Ενεργοποιείται **μετά** την αύξηση του EIP
- Επιτυγχάνεται με αλλαγή της τιμής του TF στο καταχωρητή κατάστασης
- Ο επεξεργαστής αποθηκεύει στη stack τις απαραίτητες πληροφορίες
- Καλείται η κατάλληλη διαδικασία (ρουτίνα trap handler)
- Παραδείγματα :
  - Υπερχείλιση (Overflow exception - interrupt 4)
  - Διακοπές οριζόμενες από τον προγραμματιστή (INT n)

Faults - Traps

- Είναι **σύγχρονες** διακοπές, προκαλούνται από εντολές του πργράμματος
- Ο επεξεργαστής αντιδρά με παρόμοιο τρόπο αλλά το αποτέλεσμα είναι διαφορετικό
  - Fault : Η διεύθυνση επιστροφής είναι η διεύθυνση της εντολής που προκάλεσε το σφάλμα. Μετά την αντιμετώπιση του προβλήματος η εντολή ξανα-εκτελείται.
  - Trap : Η διεύθυνση επιστροφής είναι η διεύθυνση της επόμενης εντολής.

Aborts

- Λάθος κωδικός εντολής (invalid opcode)
- Πρόβλημα hardware
- Πρόβλημα σε πίνακες του συστήματος

32-bit Interrupt – Exception Vectors

- 0 ... 19 Exceptions
- 20 ... 31 Δεσμευμένες
- 32 ... 255 Εξωτερικό σήμα διακοπής (INTR) ή διακοπή από εντολή INT n

- Interrupt and Exception Handling in Real-Address Mode
  - Ο αριθμός της διακοπής ή της εξαίρεσης χρησιμοποιείται ως για να βρεθεί από τον πίνακα IVT ο δείκτης (CS:EIP) για την ρουτίνα της διακοπής (ISR)
- INT n (0 ... 224) User defined interrupts
- INT 3 : Break point exception (debuger)
- INTO : Overflow interrupt

Διακοπές και εξαιρέσεις σε 64 bit
Interrupt and Exception Behavior in 64-Bit Mode

- Ολες οι ρουτίνες ISR είναι 64bit
- Η stack είναι 64 bit. Αν η τιμή που θα αποθηκευθεί είναι 32,16 τα υπόλοιπα bit είναι μηδέν
- Ο δείκτης της stack (SS:RSP) αποθηκεύεται οταν γίνει διακοπή και χωρίς αλλαγή επιπέδου προνομίων
- Ο καταχωρητής SS είναι όταν αλλάζει το επίπεδο
- Αλλαγή στη συμπεριφορά της IRET
- Αλλαγή στη διαδικασία αλλαγής stack
-

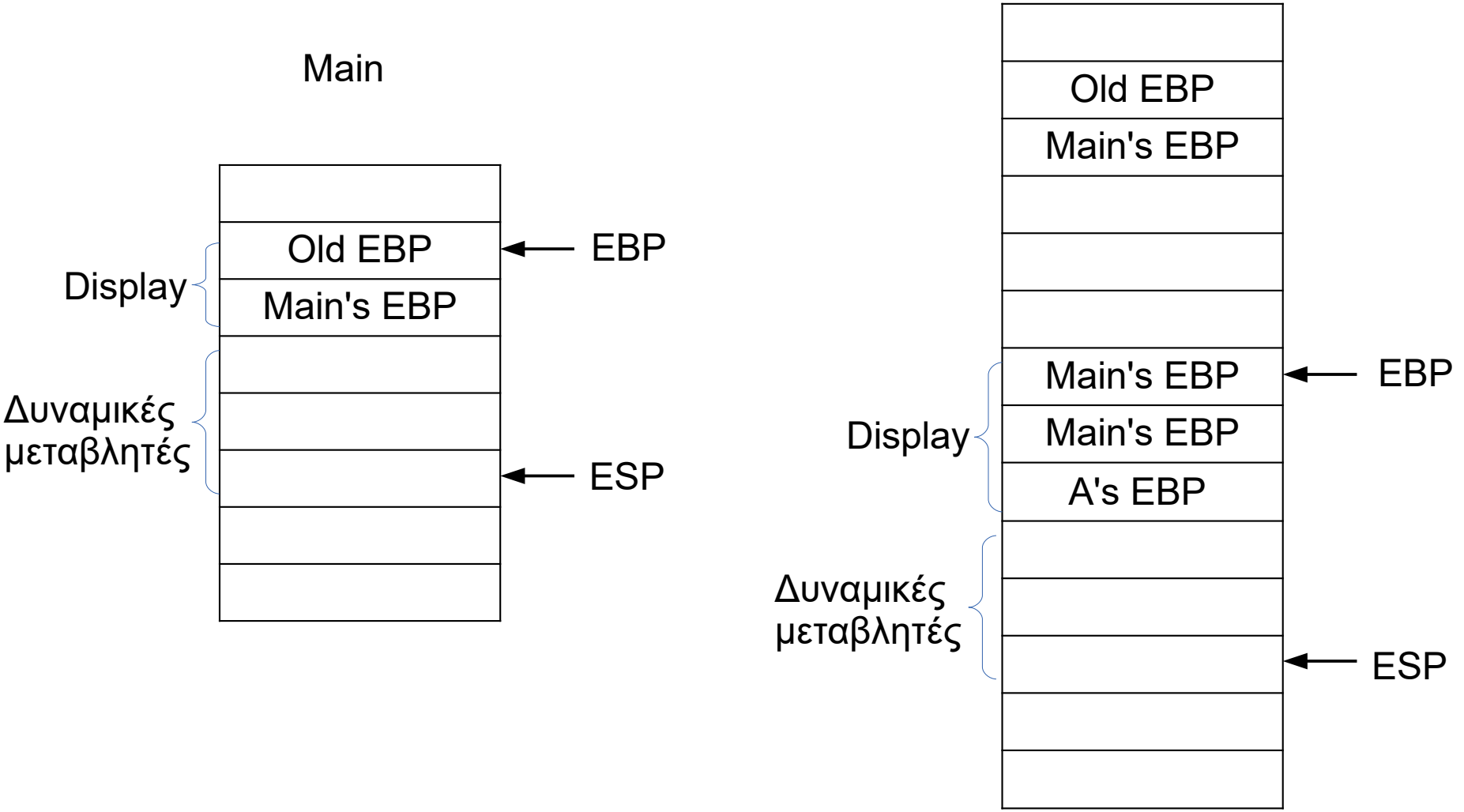
Υποστήριξη ανωτέρων γλωσσών

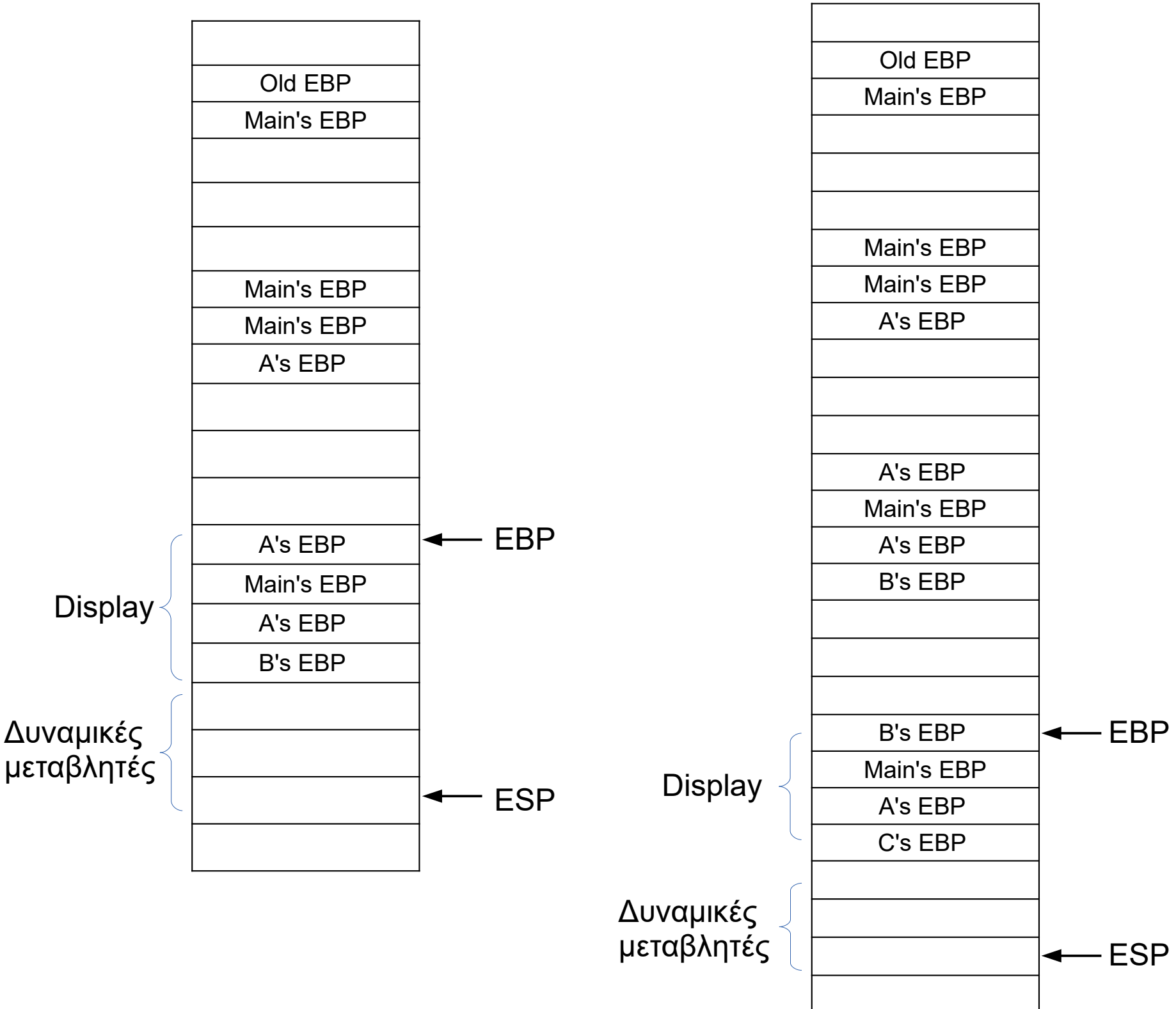
```
function_B()                Lexical Level 3
{   int fb_1,fb_2,fb_3;
    .............
    function_C();
}

function_A()                Lexical Level 2
{   int fa_1,fa_2,fa_3;
    .............
    function_B();
}

function_main()             Lexical Level 1
{
    int m_1,m_2,m_3;
    ..............
    function_A();
    ...............
}
```

- Εντολή ENTER
- ENTER m,n
- m : Μνήμη (bytes) της stack που δεσμεύεται για τοπικές (δυναμικές) μεταβλητές της διαδικασίας που καλείται.
- n (Lexical Level): 0-31. Το επίπεδο της διαδικασίας στην ιεραρχία

```
PUSH EBP;
FRAME_PTR ← ESP;
IF LEVEL > 0
   THEN
      DO (LEVEL − 1) times
            EBP ← EBP − 4;
           PUSH Pointer(EBP); (* doubleword pointed to by EBP *)
      OD;
   PUSH FRAME_PTR;
FI;
EBP ← FRAME_PTR;
ESP ← ESP − STORAGE;
```

Stack frame στην αρχή κάθε διαδικασίας

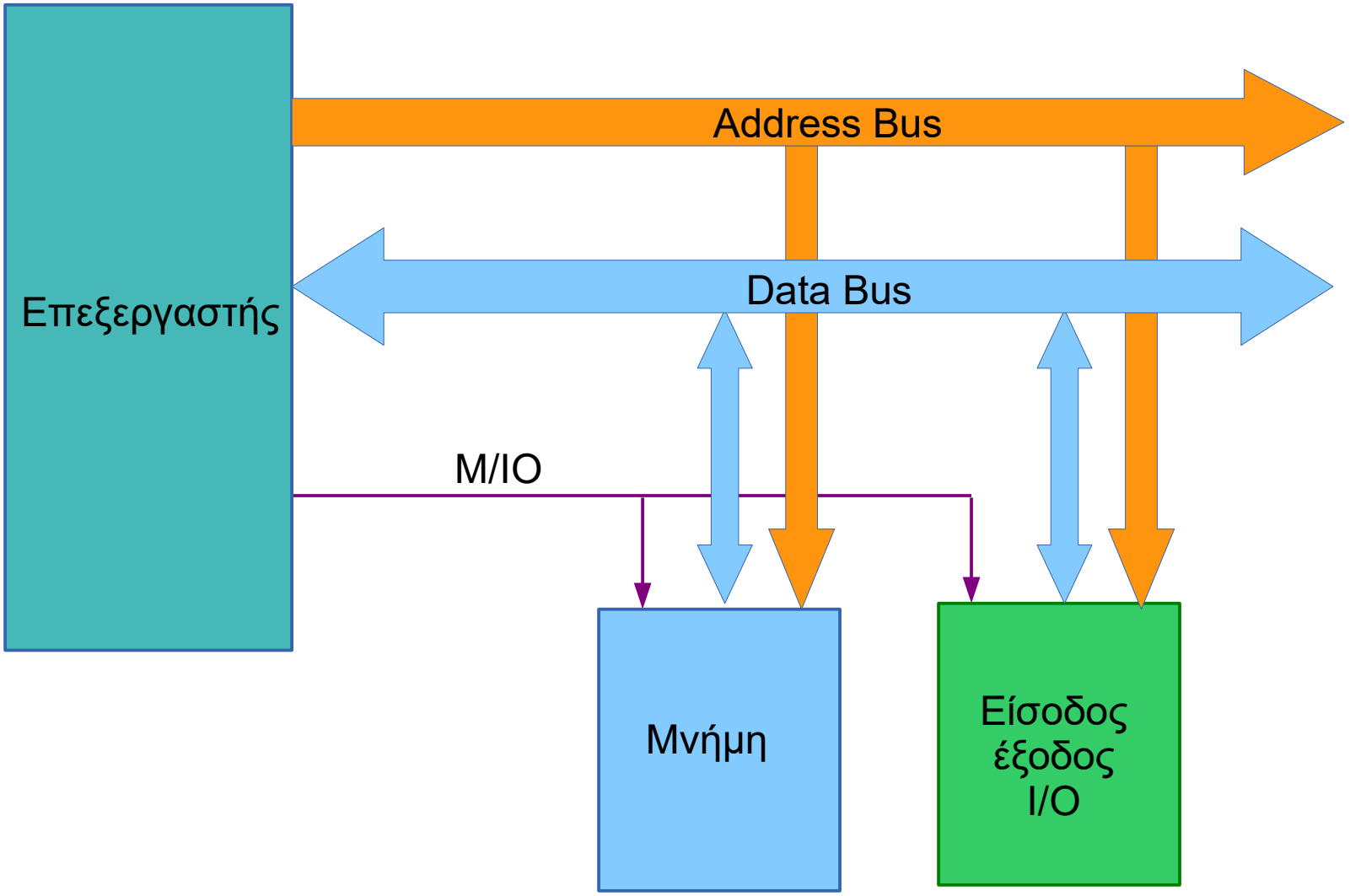



Old EBP
Main's EBP
Main's EBP
Main's EBP
A's EBP
A's EBP
Main's EBP
A's EBP
B's EBP
EBP
ESP
Display
Δυναμικές μεταβλητές
Old EBP
Main's EBP
Main's EBP
Main's EBP
A's EBP
A's EBP
Main's EBP
A's EBP
B's EBP
B's EBP
Main's EBP
A's EBP
C's EBP
EBP
ESP
Display
Δυναμικές μεταβλητές

- Η A έχει πρόσβαση μόνο στις μεταβλητές της κύριας
- Η B έχει πρόσβαση μόνο στις μεταβλητές της κύριας και της A, όχι στις μεταβλητές των C και D
- Η C έχει πρόσβαση μόνο στις μεταβλητές της κύριας και της A, όχι στις μεταβλητές των B και D
- Η D έχει πρόσβαση μόνο στις μεταβλητές της κύριας της A και της C, όχι στις μεταβλητές της B

INPUT/OUTPUT

Πρόσβαση σε θύρες είσόδου-εξόδου μέσω ειδικής περιοχής διευθύνσεων (I/O address space)



Πρόσβαση σε θύρες εισόδου-εξόδου μέσω διευθύνσεων μνήμης (Memory Mapped I/O)

- 64KB διευθύνσεις I/O
- Μπορούν να χρησιμοποιηθούν ως 8,16,32 bit ports
  - 64K 8-bit
  - 32K 16-bit Διευθύνσεις πολλαπλάσια του 2
  - 16K 32-bit Διευθύνσεις πολλαπλάσια του 4

Εντολές I/O

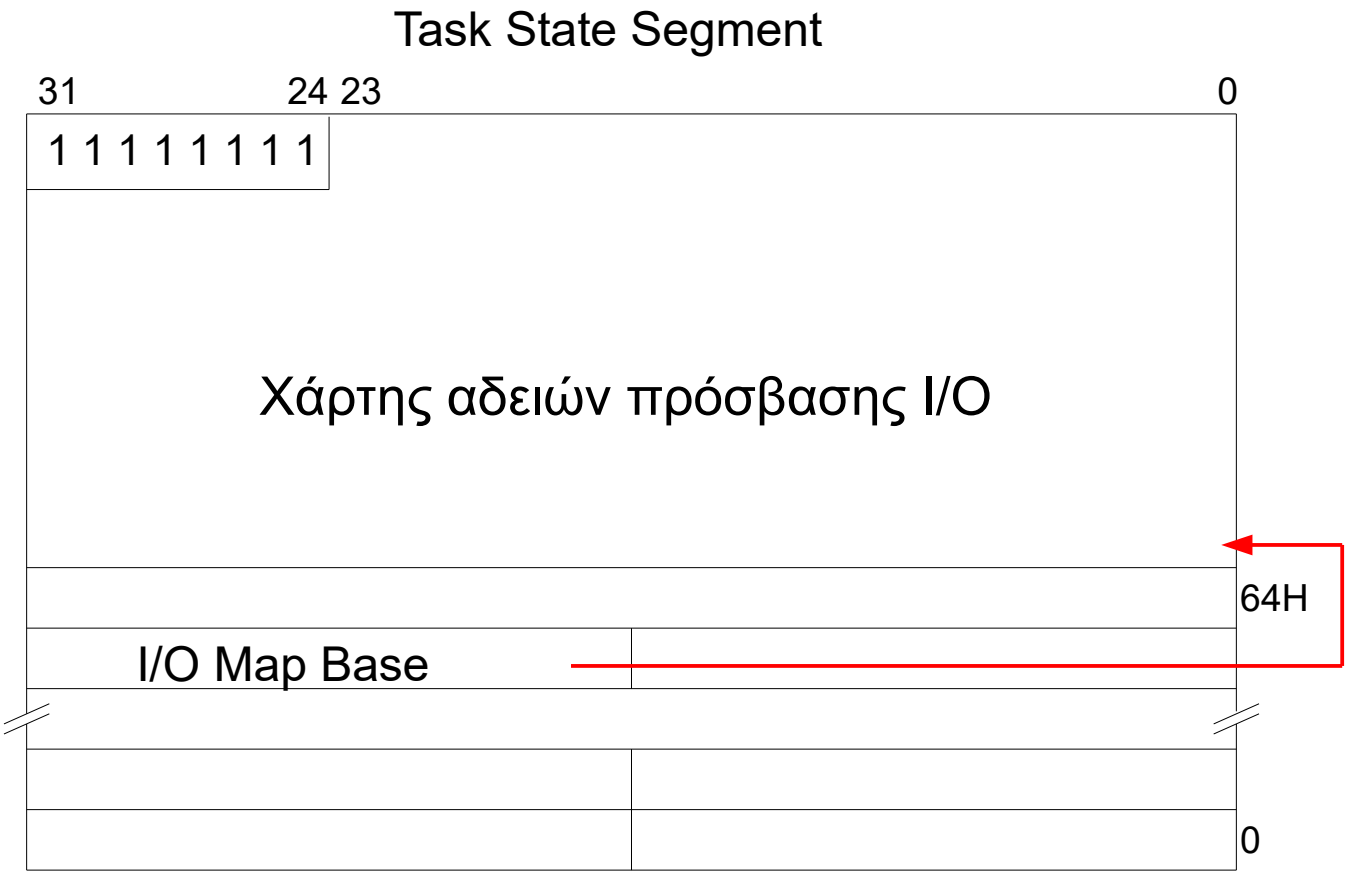
- IN , OUT
  - IN AL,port8 port8<256
  - IN AL,DX
- INS , OUTS
  - INS : ES:(E)SI buffer
  - OUTS : DS:(E)DI buffer
- Η εντολή OUT ολοκληρώνει την εγγραφή των δεδομένων **πριν** εκτελεστεί η επόμενη εντολή

Memory-Mapped I/O

- Οταν η περιφερειακή συσκευή απεικονίζεται σαν μνήμη πρέπει να απενεργοποιείται η cache

PROTECTED-MODE I/O

- Για να χρησιμοποιήσει ένα πρόγραμμα εντολές I/O πρέπει να έχει τα κατάλληλα προνόμια που καθορίζονται από
  - Δύο bits (I/O Privilege Level, IOPL) του καταχωρητή EFLAGS
  - Χάρτης άδειας πρόσβασης (I/O permitipn map) του TSS
- Ο πυρήνας του λειτουργικού συστήματος και τα device drivers έχουν πρόσβαση στο χώρο διευθύνσεων I/O
- Το IOPL μπορεί να αλλάξει μόνο από προγράμματα που τρέχουν στο επίπεδο προνομίων 0
- Οι εφαρμογές πρέπει να χρησιμοποιούν τις υπηρεσίες του λειτουργικού συστήματος



- Ο χάρτης μπορεί να μην υπάρχει, τότε το πρόγραμμα δεν επιτρέπεται να χρησιμοποιεί εντολές I/O
- Κάθε bit αντιστοιχεί σε ένα I/O port 8 bit (μπορεί να μην υπάρχουν όλα).
- Εάν CPL > IOPL ο επεξεργαστής εξετάζει τα bits του χάρτη και αν είναι '1' για κάθε port για το οποίο ζητείται προσβαση τότε την επιτρέπει διαφορετικά σηματοδοτείται μια εξαίρεση (general-protection exception, #GP)