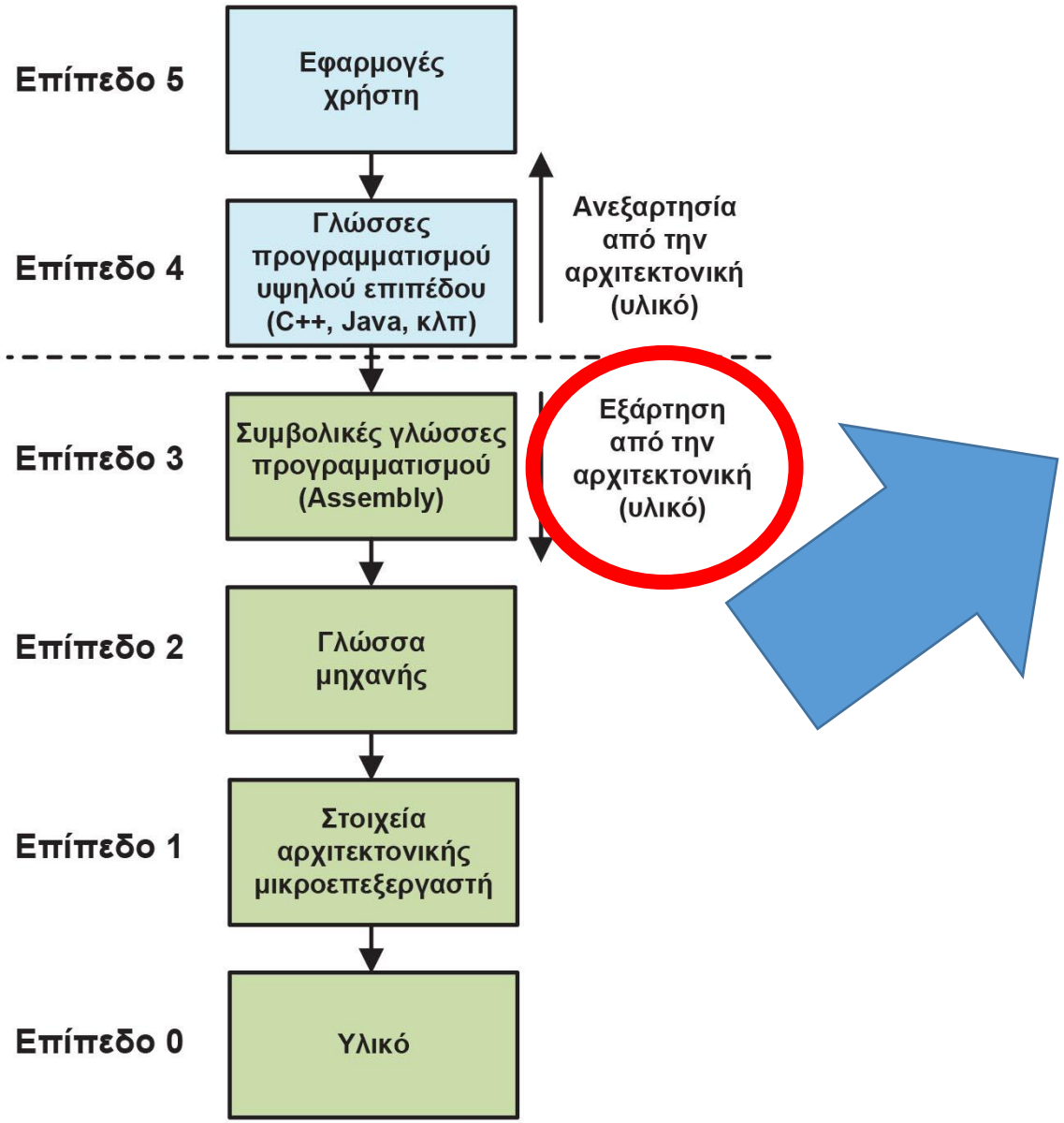
Κεφάλαιο 5

Προγραμματισμός μικροεπεξεργαστών

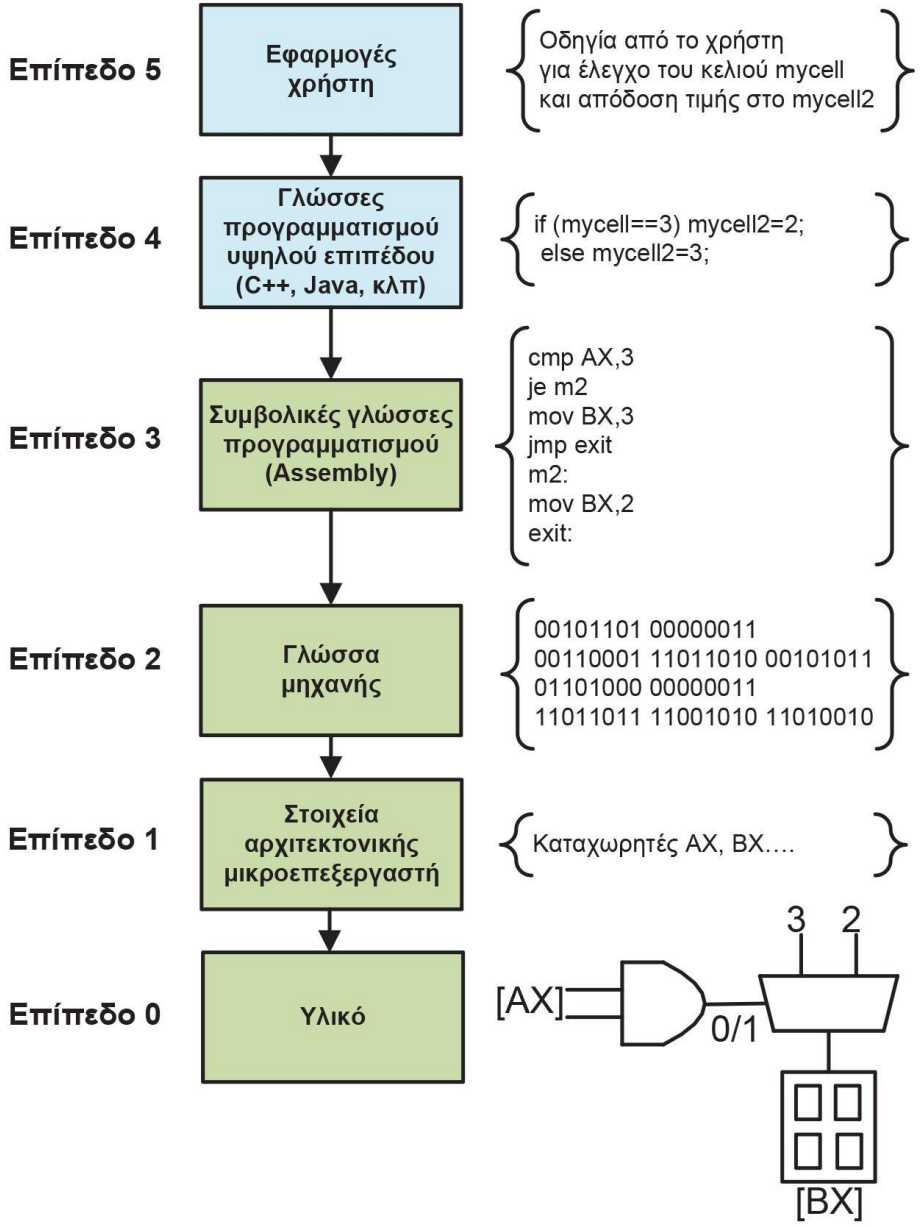
- Οι βασικές γνώσεις για να ξεκινήσουμε τον προγραμματισμό ενός μικροεπεξεργαστή, είναι οι καταχωρητές που διαθέτει, καθώς και οι αντίστοιχες εντολές Assembly που υποστηρίζει
- Με άλλα λόγια, πρέπει να γνωρίζουμε την αρχιτεκτονική του μικροεπεξεργαστή
- Η γλώσσα Assembly είναι γνωστή και ως συμβολική γλώσσα
- Η συμβολική γλώσσα αντανακλά την αρχιτεκτονική του μικροεπεξεργαστή και επομένως, κάθε μικροεπεξεργαστής έχει τη δική του (συμβολική γλώσσα)
- Ένα πρόγραμμα σε συμβολική γλώσσα δεν τρέχει σε μικροεπεξεργαστή διαφορετικής αρχιτεκτονικής (εκτός αν ανήκει στην ίδια οικογένεια, όπου διατηρούνται στοιχεία αρχιτεκτονικής για λόγους συμβατότητας)
- Όλα τα προγράμματα μετατρέπονται σε συμβολική γλώσσα πριν το τελικό στάδιο εκτέλεσής τους από τον μικροεπεξεργαστή

Σε αυτή την παρουσίαση θα αναλύσουμε σύντομα τον προγραμματισμό του μικροεπεξεργαστή 8086/8088 (αρχιτεκτονική INTEL 16bit)

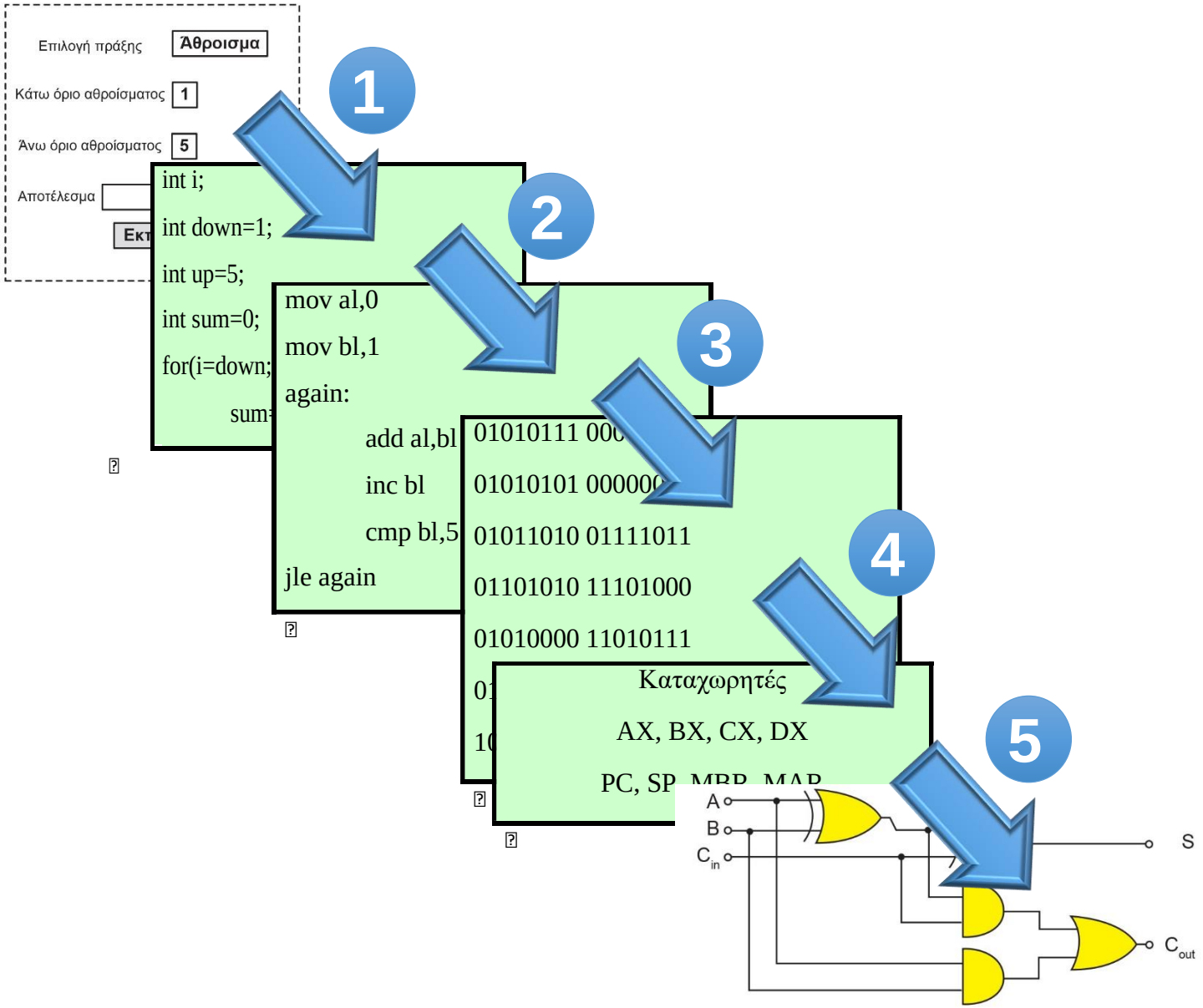
Επίπεδο γλώσσας Assembly



Παράδειγμα



«Ροή» υλοποίησης οδηγίας-εντολής



Γιατί Assembly ;

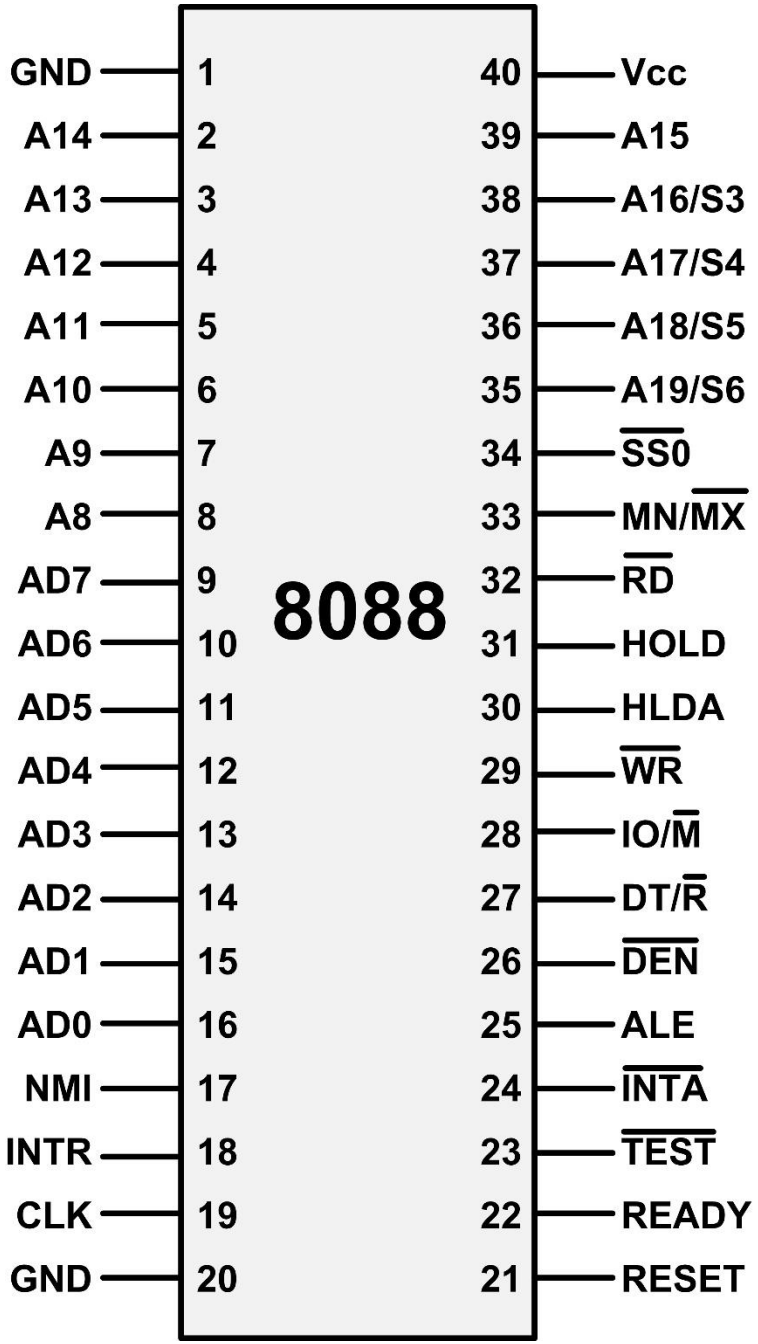
- Πρόσβαση σε όλες τις δυνατότητες του μικροεπεξεργαστή
- Μικρότερα και ταχύτερα προγράμματα
- «Μονόδρομος» σε απαιτητικούς υπολογισμούς (π.χ. παιχνίδια)
- Άμεση πρόσβαση στο υλικό
- Κατανόηση της αρχιτεκτονικής

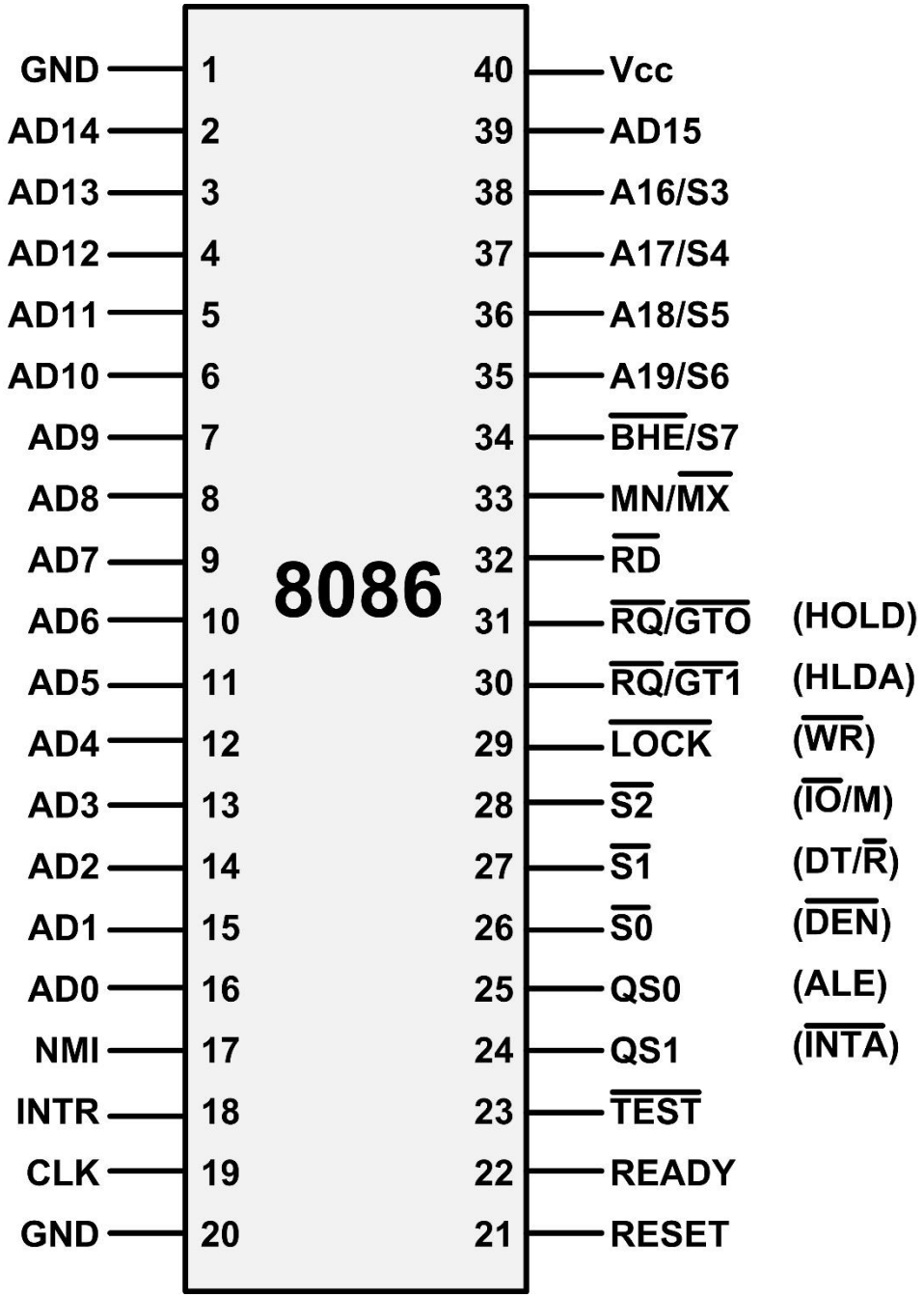
Μειονεκτήματα

- Υψηλή πολυπλοκότητα
- Μεγαλύτερος χρόνος ανάπτυξης
- Μη μεταφέρσιμος κώδικας

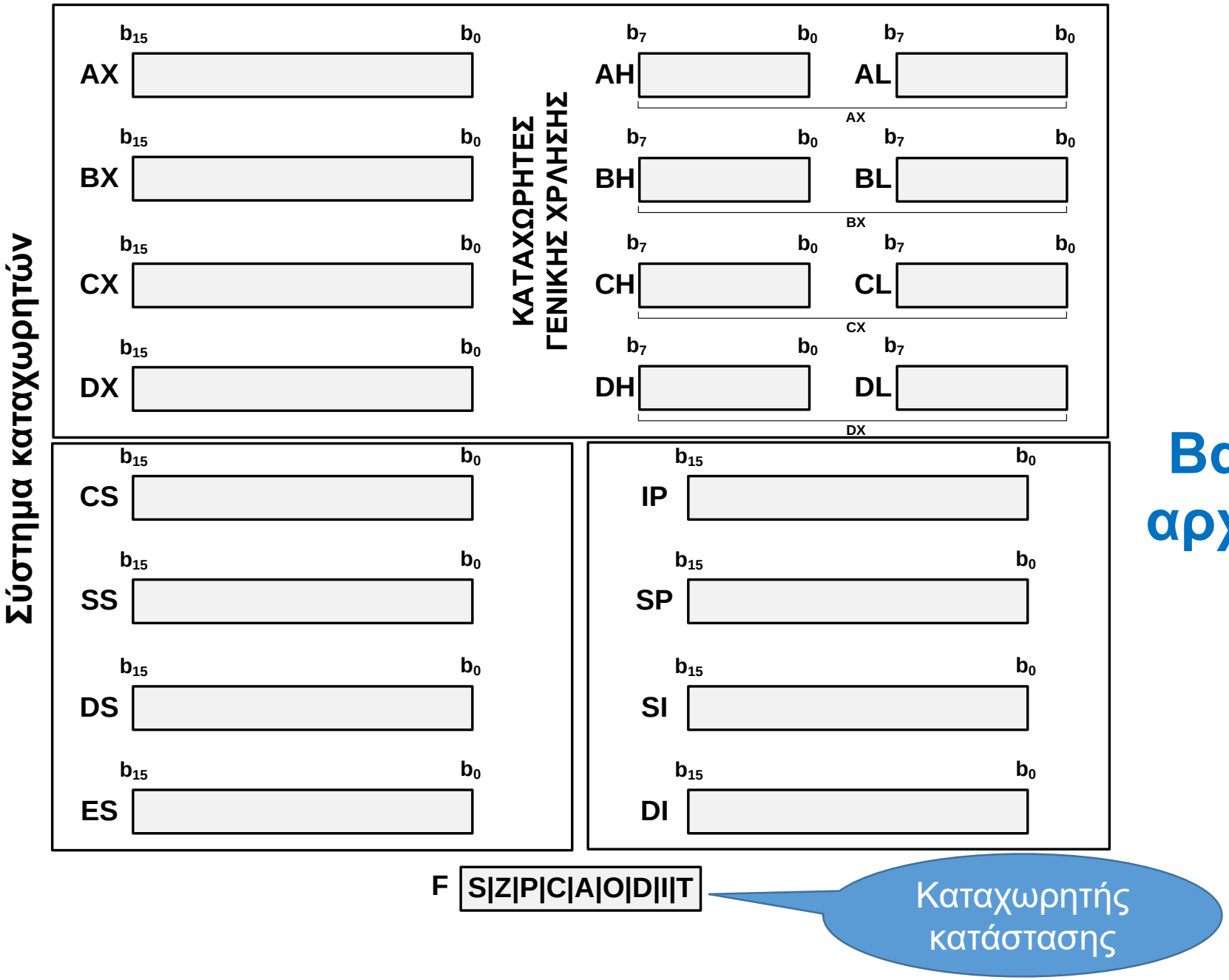
Γιατί 8086 ;

- **Για να μάθουμε τη γενική φιλοσοφία προγραμματισμού σε συμβολική γλώσσα**
- **Για να δούμε έναν επεξεργαστή φιλοσοφίας CISC**
- **Για να μπορούμε να «διαβάζουμε» και να «μελετάμε» πολυπλοκότερους μικροεπεξεργαστές αξιοποιώντας το αντίστοιχο εγχειρίδιο**
- **Οι σημερινοί μικροεπεξεργαστές είναι εξαιρετικά πολύπλοκοι για να εισαχθούν στην εκπαιδευτική διαδικασία, ενώ η αντίστοιχη γνώση θα ήταν αυστηρά εξειδικευμένη**


GND 1
A14 2
A13 3
A12 4
A11 5
A10 6
A9 7
A8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
8088
40 Vcc
39 A15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 SS0
33 MN/MX
32 RD
31 HOLD
30 HLDA
29 WR
28 IO/M
27 DT/R
26 DEN
25 ALE
24 INTA
23 TEST
22 READY
21 RESET

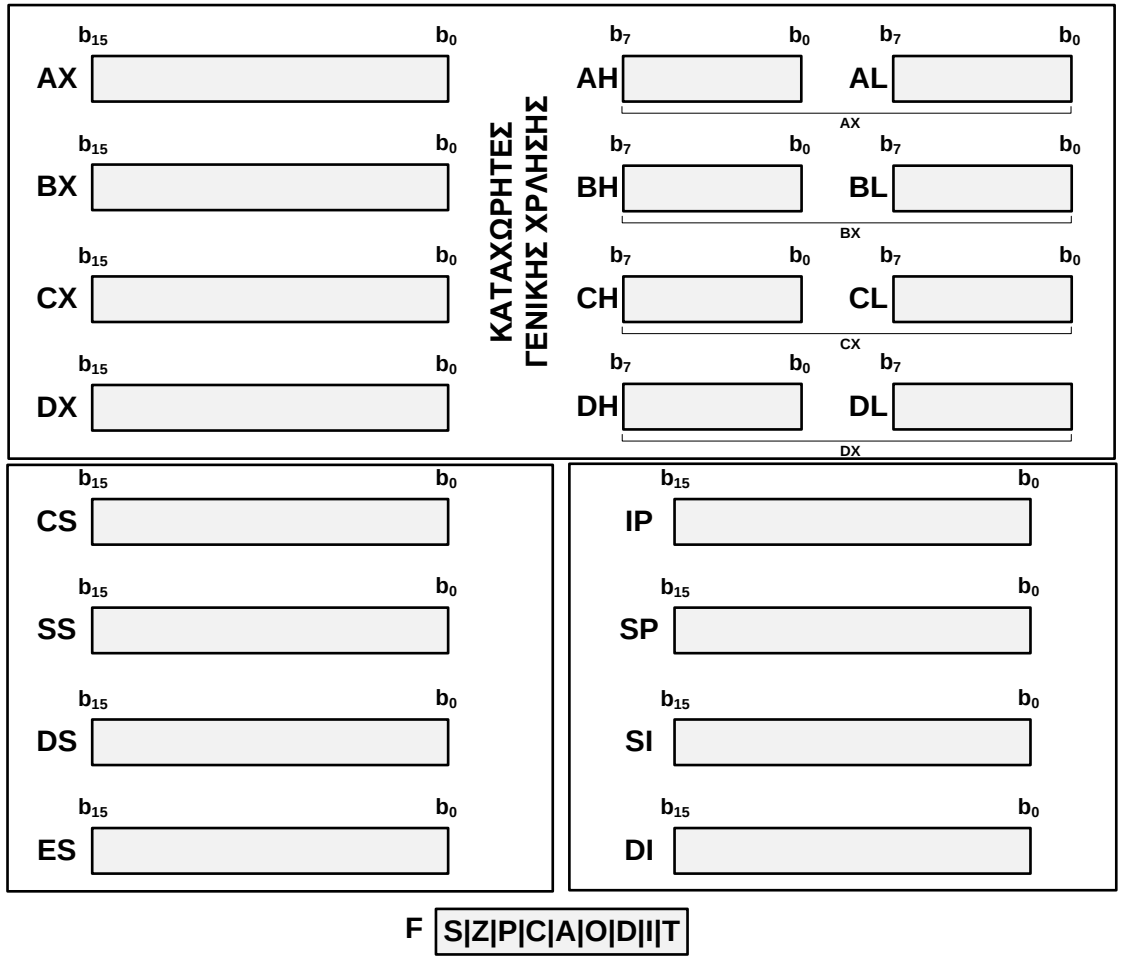

GND 1
AD14 2
AD13 3
AD12 4
AD11 5
AD10 6
AD9 7
AD8 8
AD7 9
AD6 10
AD5 11
AD4 12
AD3 13
AD2 14
AD1 15
AD0 16
NMI 17
INTR 18
CLK 19
GND 20
8086
40 Vcc
39 AD15
38 A16/S3
37 A17/S4
36 A18/S5
35 A19/S6
34 BHE/S7
33 MN/MX
32 RD
31 RQ/GT0 (HOLD)
30 RQ/GT1 (HLDA)
29 LOCK (WR)
28 S2 (IO/M)
27 S1 (DT/R)
26 S0 (DEN)
25 QS0 (ALE)
24 QS1 (INTA)
23 TEST
22 READY
21 RESET


Δρ. Π. Παπάζογλου, Προγραμματισμός σε συμβολική Γλώσσα



Βασικά συστατικά αρχιτεκτονικής 16bit

Καταχωρητές



- Διαθέτει τέσσερις βασικούς καταχωρητές γενικής χρήσης 16bit (AX, BX, CX, DX), που μπορούν να «διαιρεθούν» σε δύο των 8bit.
- Για παράδειγμα, ο καταχωρητής AX μπορεί να «διαιρεθεί» σε AH (high byte) και AL (low byte) και έτσι, να διπλασιάσουμε τον αριθμό τους.
- Αλλάζοντας το περιεχόμενο των AH ή AL, αμέσως αλλάζει συνολικά και το περιεχόμενο του AX.

Καταχωρητής κατάστασης (μερικά βασικά Flag)

SF (Sign Flag)
Ανίχνευση bit προσήμου στο αποτέλεσμα μιας πράξης

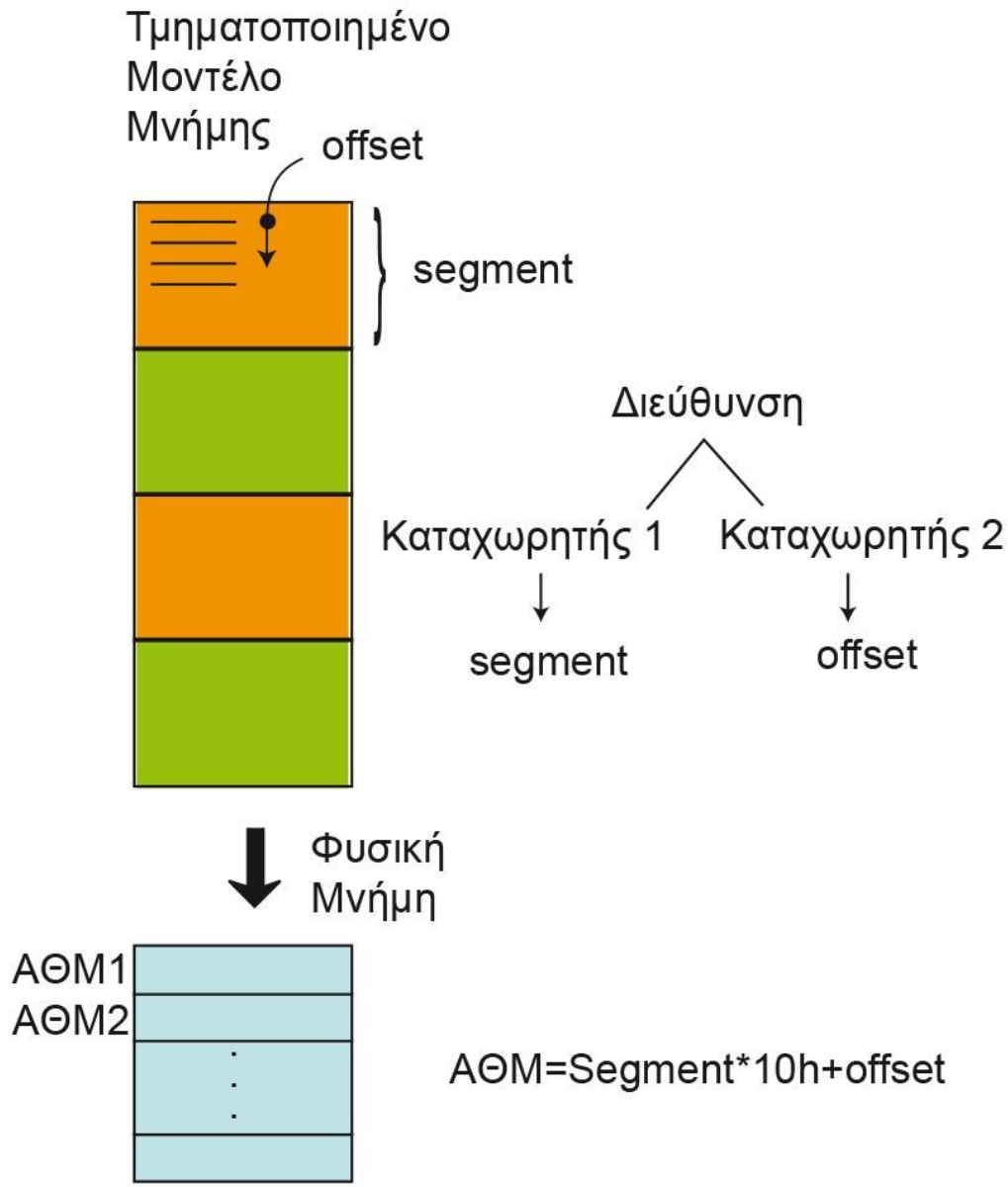
ZF (Zero Flag)
Αν το αποτέλεσμα κάποιας πράξης είναι μηδέν, τότε αυτό το bit ενεργοποιείται (τιμή 1)

CF (Carry Flag)
Αν κάποια πρόσθεση παράγει κρατούμενο πέρα και από το περισσότερο σημαντικό ψηφίο, τότε αυτό το bit ενεργοποιείται. Επίσης, παίρνει και την τιμή 1 αν απαιτηθεί δανεικό σε περίπτωση αφαίρεσης.

IF (Interrupt Flag)
Το bit αυτό δείχνει αν ο μικροεπεξεργαστής θα ανιχνεύει τυχόν διακοπές ή όχι

TF (Trap Flag)
Αυτό το ψηφίο χρησιμοποιείται σε περιπτώσεις που επιθυμούμε να γίνει εκτέλεση προγράμματος εντολή προς εντολή

Οργάνωση μνήμης σε τμήματα (1)



Λόγω των 16bit καταχωρητών, απαιτούνται δύο καταχωρητές για την προσπέλαση ολόκληρης της μνήμης. Το πρόβλημα λύθηκε με την οργάνωση της μνήμης σε τμήματα. Έτσι τώρα, για να αναφερθούμε σε μια θέση μνήμης, θα πρέπει να γνωρίζουμε τόσο το τμήμα, όσο και τη θέση μέσα σε αυτό.

- INTEL 16bit = μνήμη οργανωμένη σε τμήματα (segment) των 64 Kbyte
- Θέση μέσα σε τμήμα = offset
- Προσδιορισμός θέσης στη μνήμη = segment:offset
- Στη φυσική μνήμη ο χώρος είναι ενιαίος και κάθε θέση μνήμης προσδιορίζεται με μια μόνο διεύθυνση (Απόλυτη Θέση Μνήμης – ΑΘΜ)

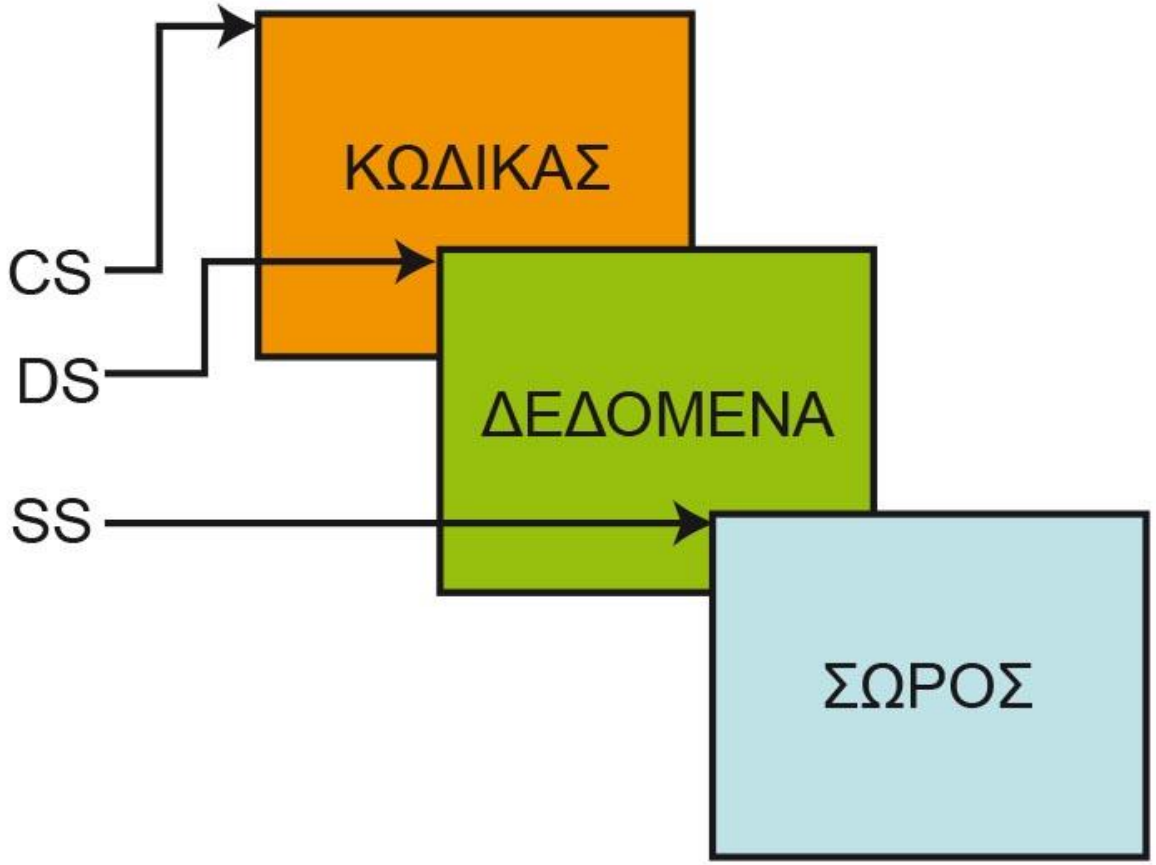
Οργάνωση μνήμης σε τμήματα (2)

CS (Code Segment)
- Διεύθυνση τμήματος κώδικα

DS (Data Segment)
- Διεύθυνση τμήματος δεδομένων (που διαχειρίζεται το πρόγραμμα)
- Σε πολλές περιπτώσεις το περιεχόμενο αυτών των δύο καταχωρητών ταυτίζεται (κώδικας και δεδομένα στο ίδιο τμήμα)

SS (Stack Segment)
- Διεύθυνση τμήματος σωρού

ES (Extra Segment)
- Καταχωρητής τμήματος (π.χ. για επεξεργασία αλφαριθμητικών)



Μερικές εντολές …

Φόρτωση καταχωρητή με ακέραιο αριθμό

π.χ. **MOV AL,09**

Εκτελεί την εκχώρηση AL=9

Μείωση κατά 1

π.χ. **DEC AL**

Εκτελεί την πράξη AL=AL-1

Άλμα υπό συνθήκη

π.χ. **JNZ Start**

Άλμα στη διεύθυνση Start, όσο ZF=0. Αν το αποτέλεσμα της προηγούμενης πράξης είναι μηδέν, τότε ZF=1 και η εκτέλεση συνεχίζεται από την επόμενη εντολή (δεν γίνεται άλμα). Για παράδειγμα, όταν η εντολή DEC AL, δώσει AL=0, τότε ZF=1

Υλοποίηση αλγόριθμου (1)

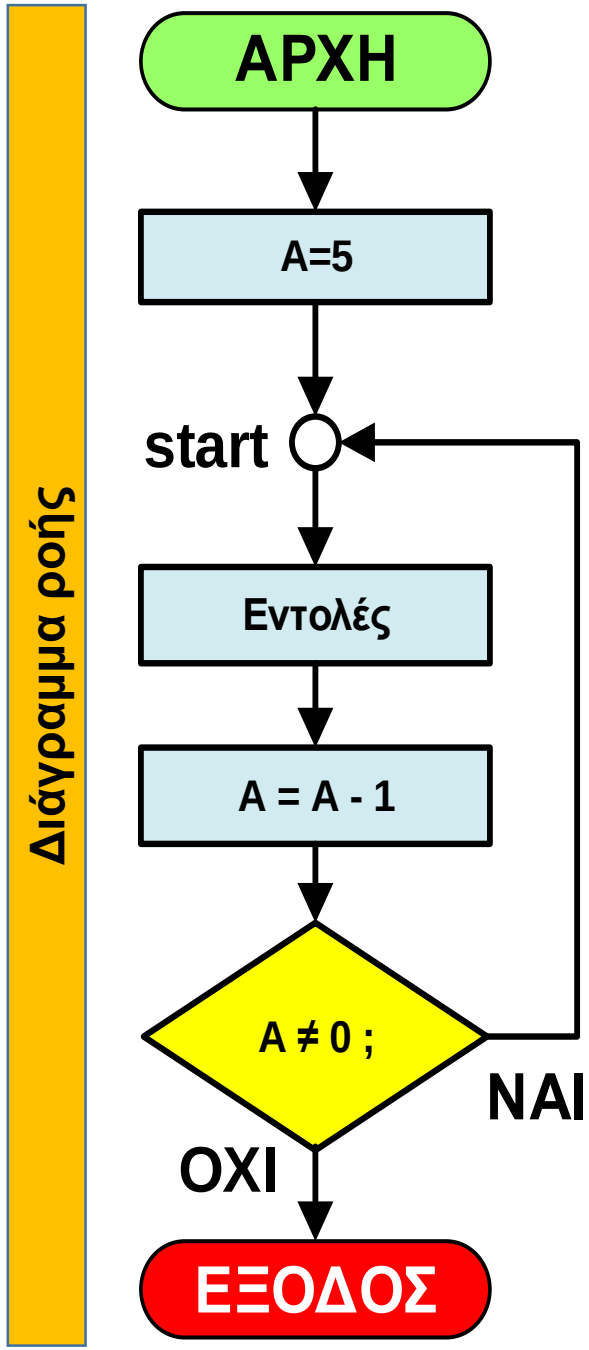
- **Επιθυμούμε να υλοποιήσουμε ένα βρόχο επανάληψης της μορφής do-while σε Assembly, όμως δεν γνωρίσουμε σχεδόν τίποτα από Assembly**

- **Γνωρίζουμε όμως τους βασικούς αλγορίθμους**

- **Έστω Α ο μετρητής του βρόχου με τιμές 5 έως 0, ως εξής:**

```
Α=5
Επανέλαβε
        Εντολές που θέλουμε να επαναλαμβάνονται
        Α=Α-1
Όσο Α≠0
```

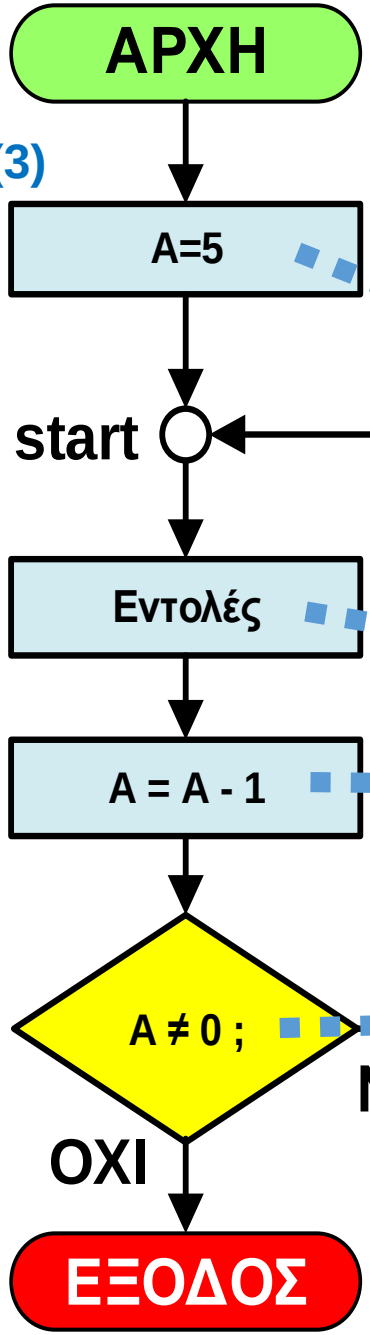
Υλοποίηση αλγόριθμου (2)




```
A=5
Επανέλαβε
        Εντολές που θέλουμε να επαναλαμβάνονται
        A=A-1
Όσο A≠0
```

Υλοποίηση αλγόριθμου (3)

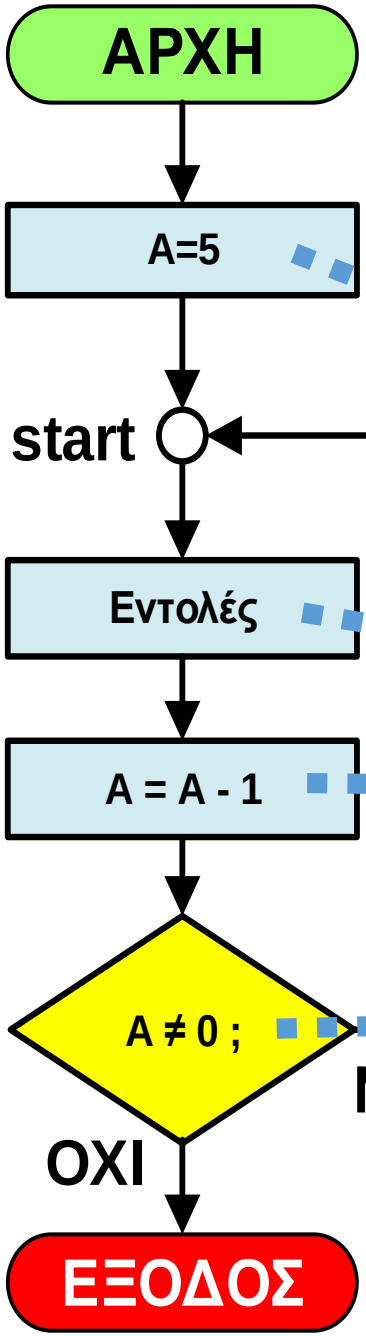
Υλοποίηση σε γλώσσα C



Κώδικας C

```
int A=5;
do
 {

//Εντολές
A=A-1;

}
while (A!=0)
```

Υλοποίηση αλγόριθμου (4)



Κώδικας C

```
int A=5;
do
 {

 //Εντολές
 A=A-1;

}
while (A!=0)
```

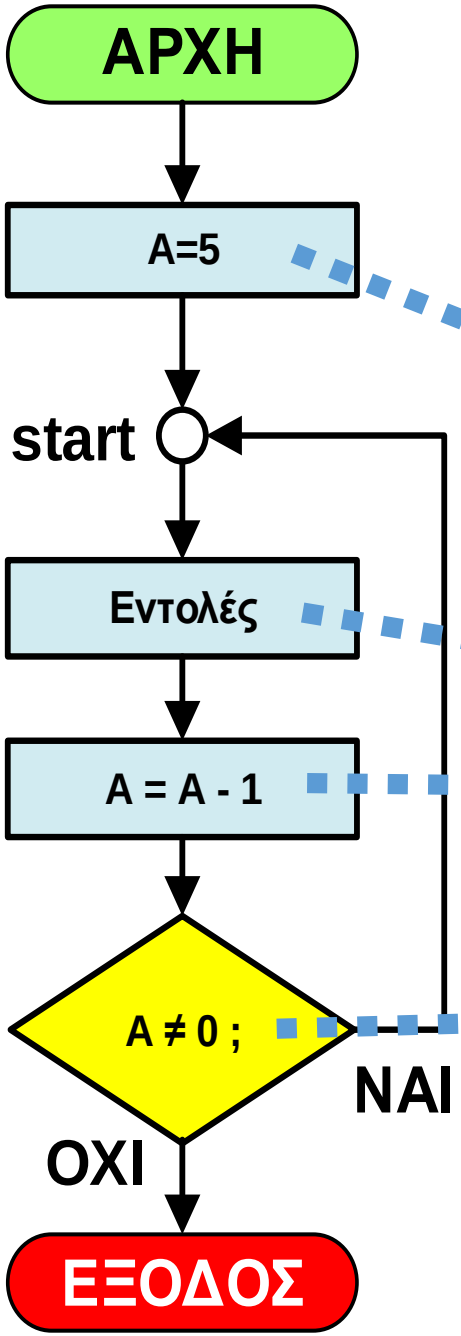
Κώδικας Assembly

```
MOV AL,5
start:

 ;Εντολές
 DEC AL

JNZ start
```

Ποια κοινά βρίσκετε ;

Τα πάντα είναι αλγόριθμος …

Υλοποίηση αλγόριθμου (5)



Κώδικας C

```
int A=5;
do
 {

 //Εντολές
 A=A-1;

 }
while (A!=0)
```

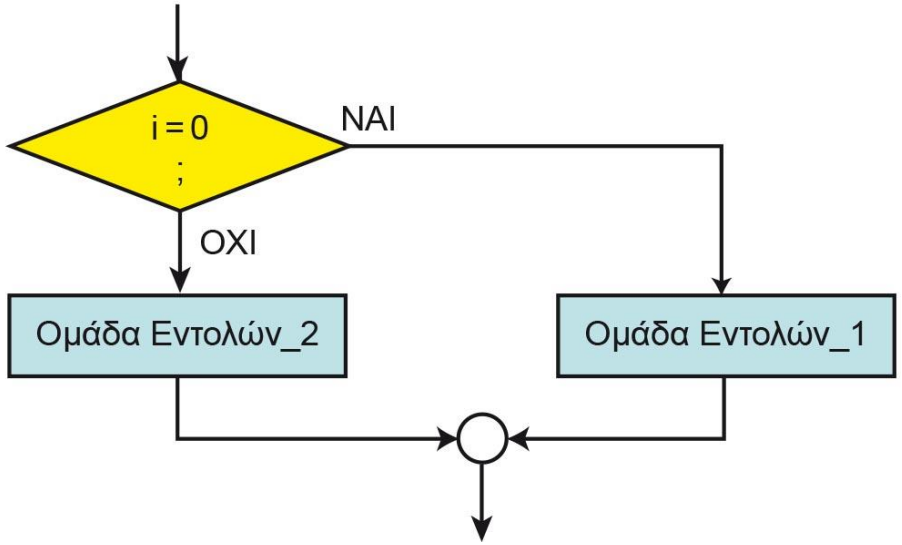
Κώδικας Assembly

```
MOV AL,5
start:

;Εντολές
DEC AL

JNZ start
```

Δεν χρειαζόταν να έχω κάποια ειδική γνώση στην Assembly…!!!

Υλοποίηση if-then-else (1)

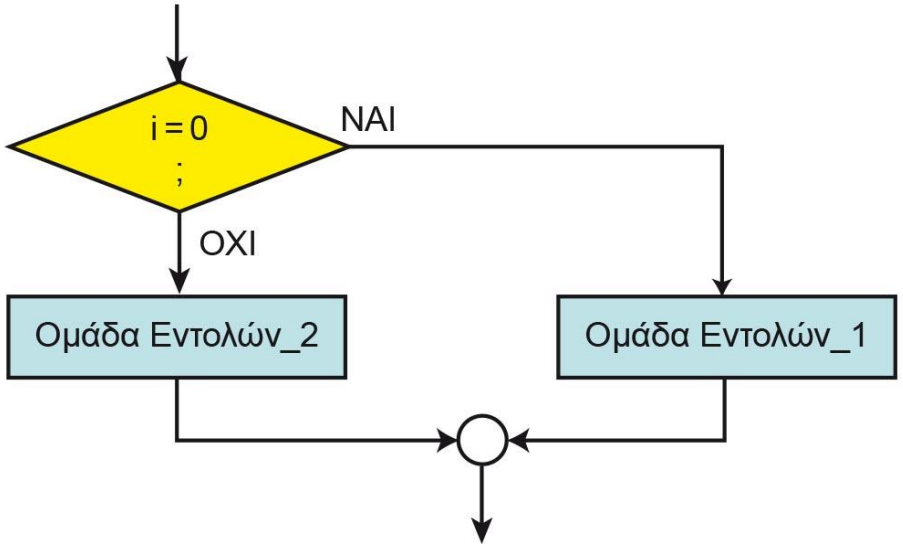
Η λογική ενός απλού ελέγχου




```
if (i==0)
        { //ομάδα εντολών_1
          //i=0
        }

else

        { //ομάδα εντολών_1
          //i≠0
        }
```

Υλοποίηση if-then-else (1)

Η λογική ενός απλού ελέγχου




```
CMP CH,0
JE TRUE_omada_1

FALSE_omada_2:
;Ομάδα_εντολών_2
;CH≠0
JMP exit

TRUE_omada_1:
;Ομάδα_εντολών_1
;CH=0

exit:
```

Υλοποίηση if-then-else (2)

```
CMP CH,0
JE TRUE_omada_1

FALSE_omada_2:
;Ομάδα_εντολών_2
;CH≠0
JMP exit

TRUE_omada_1:
;Ομάδα_εντολών_1
;CH=0

exit:
```

Βήματα υλοποίησης

1. **Έλεγχος (εντολή CMP)**
2. **Εντολή άλματος (τύπου jump) που βάσει συνθήκης οδηγεί τη ροή εκτέλεσης σε επιλεγμένο σημείο**

CMP κατ1,κατ2 ;σύγκριση περιεχομένου των καταχωρητών **κατ1** και **κατ2**

ή

CMP κατ,τιμή ;σύγκριση περιεχομένου του καταχωρητή **κατ** με την **τιμή**

Ο μικροεπεξεργαστής αφαιρεί το ένα όρισμα από το άλλο προκαλώντας την ενεργοποίηση bit από τον καταχωρητή FLAGS (π.χ. ZF=1 αν προκύψει μηδέν)

Υλοποίηση if-then-else (3)

Περισσότεροι τρόποι υλοποίησης

```
CMP CH,0
JE TRUE_omada_1

FALSE_omada_2:
;Ομάδα_εντολών_2
;CH≠0
JMP exit

TRUE_omada_1:
;Ομάδα_εντολών_1
;CH=0

exit:
```

```
CMP CH,0
JZ TRUE_omada_1

FALSE_omada_2:
;Ομάδα_εντολών_2
;CH≠0
JMP exit

TRUE_omada_1:
;Ομάδα_εντολών_1
;CH=0

exit:
```

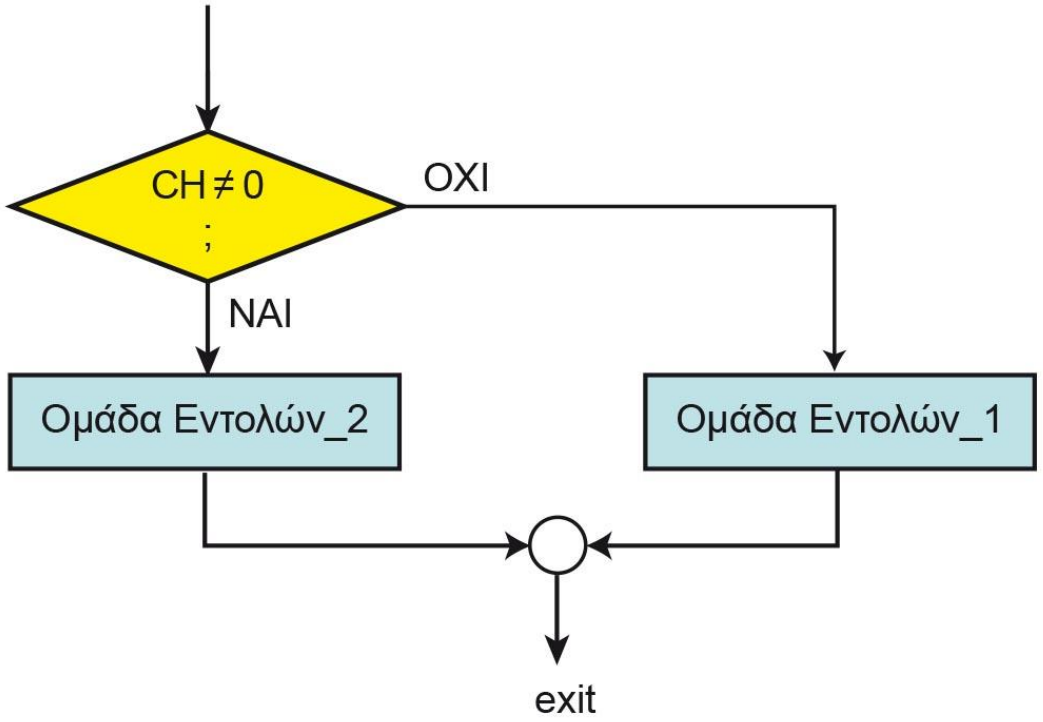
```
CMP CH,0
JNE TRUE_omada_2

FALSE_omada_1:
;Ομάδα_εντολών_1
;CH=0
JMP exit

TRUE_omada_2:
;Ομάδα_εντολών_2
;CH≠0

exit:
```

```
CMP CH,0
JNZ TRUE_omada_2

FALSE_omada_1:
;Ομάδα_εντολών_1
;CH=0
JMP exit

TRUE_omada_2:
;Ομάδα_εντολών_2
;CH≠0


exit:
```

JE=Jump Equal ή **JZ**=Jump on Zero
JNE=Jump Not Equal ή **JNZ**=Jump on Non Zero
JMP=Goto

Υλοποίηση if-then-else (4)

Έλεγχος αν CH ≠ 0

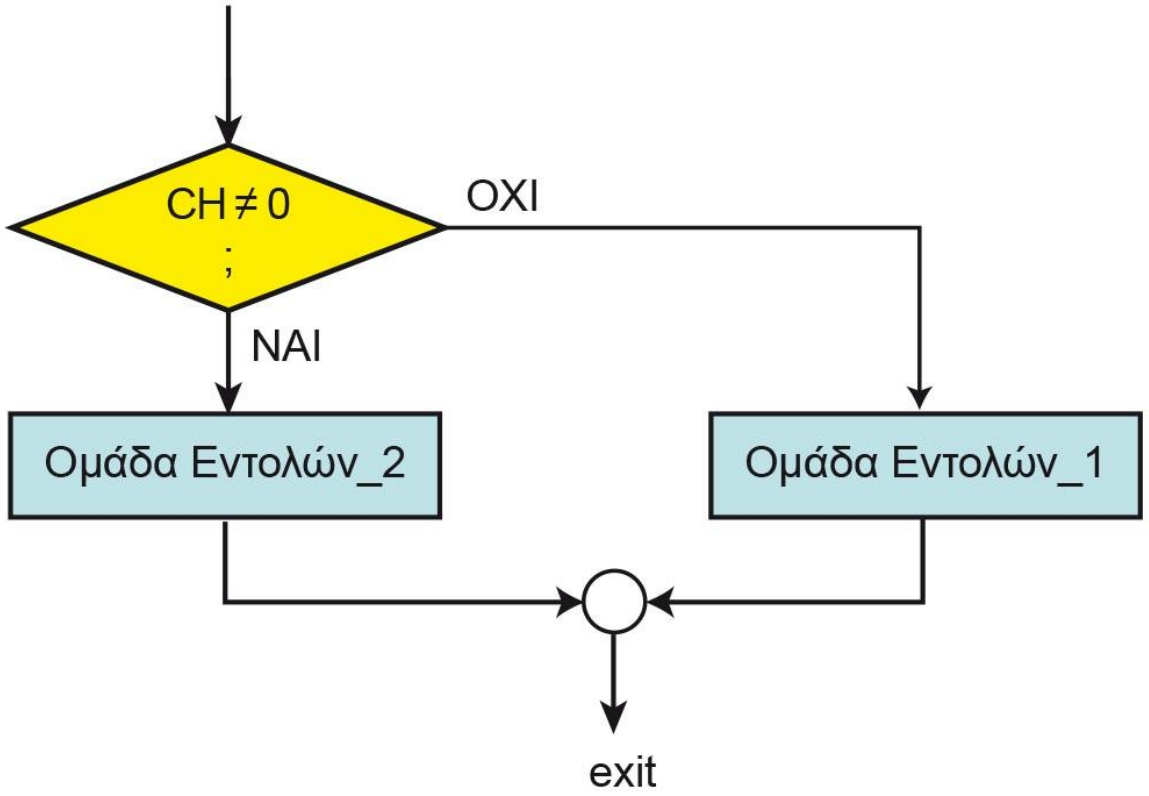


Υλοποίηση if-then-else (4)

Έλεγχος αν CH ≠ 0

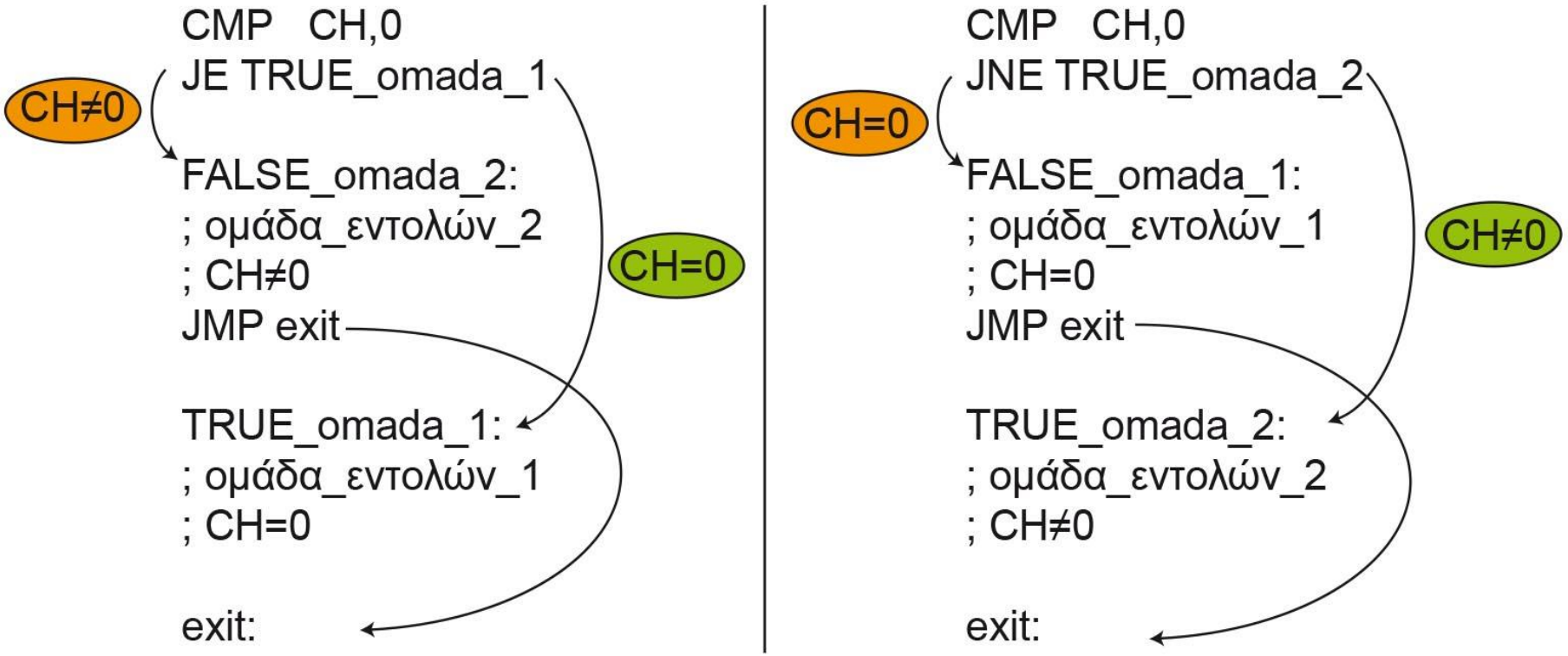
```
CMP CH,0
JNE
TRUE_omada_2

FALSE_omada_1:
;Ομάδα_εντολών_1
;CH=0
JMP exit

TRUE_omada_2:
;Ομάδα_εντολών_2
;CH≠0

exit:
```

```
CMP CH,0
JNZ
TRUE_omada_2

FALSE_omada_1:
;Ομάδα_εντολών_1
;CH=0
JMP exit

TRUE_omada_2:
;Ομάδα_εντολών_2
;CH≠0


exit:
```

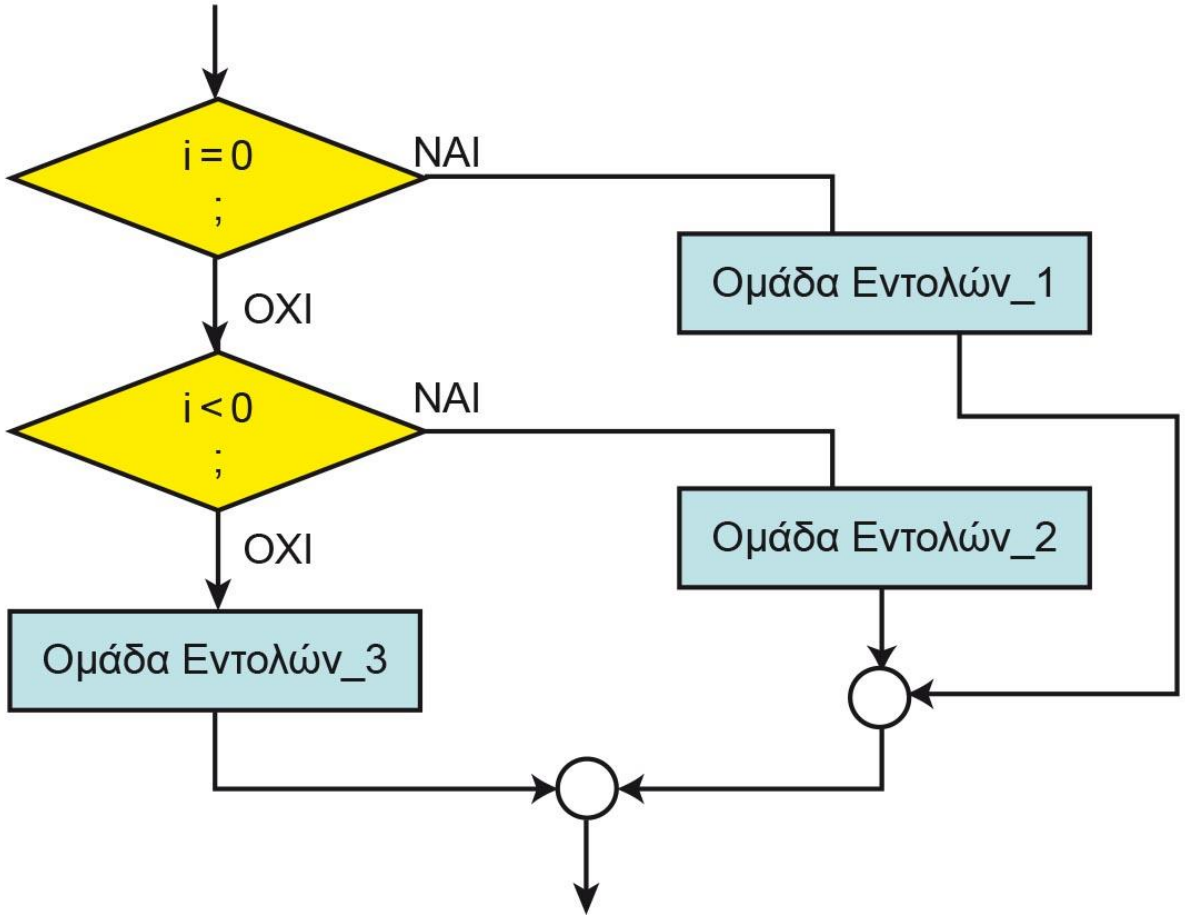


Υλοποίηση if-then-else (5)

Ροές εκτέλεσης

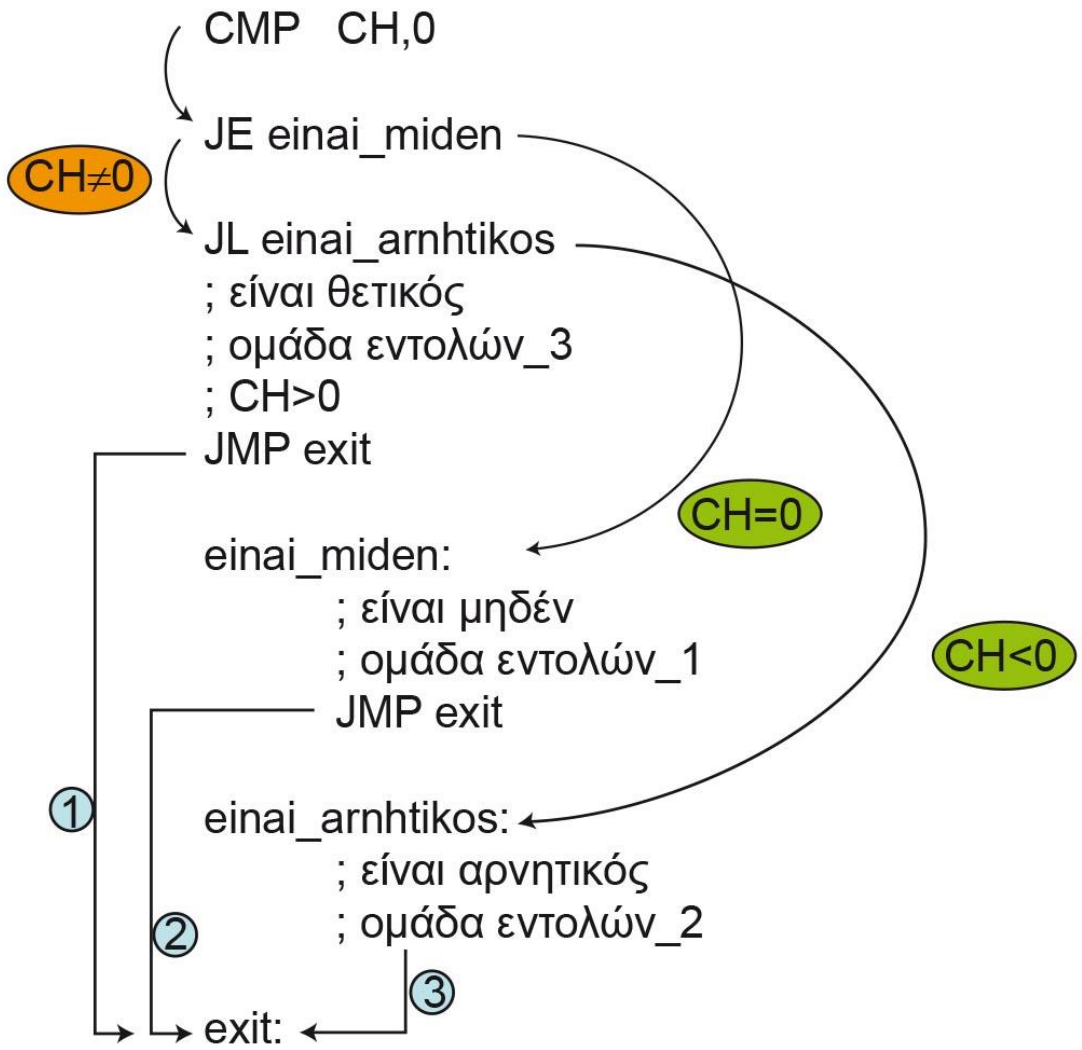


Υλοποίηση If-then-else-if (1)

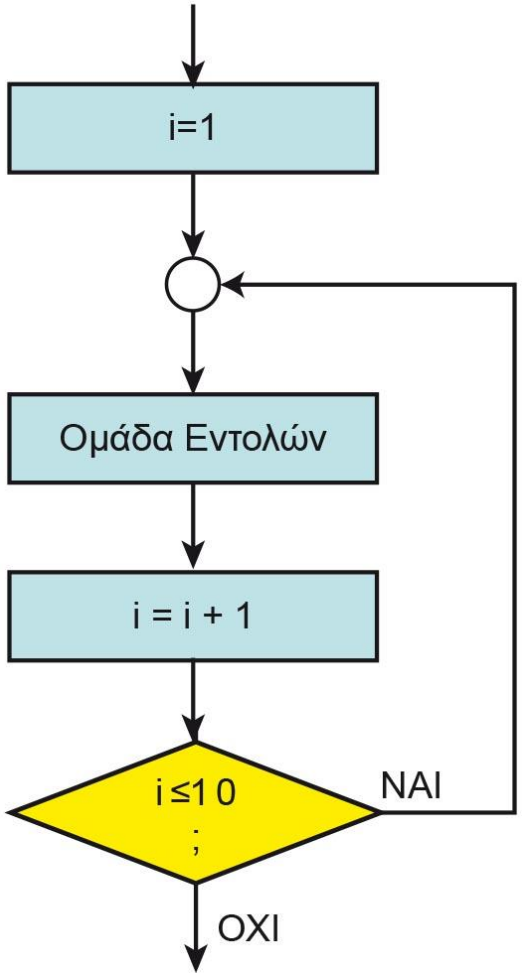



```
if (i==0)
        {//ομάδα εντολών_1
         //Μηδενικός αριθμός
        }

else
        if (i<0)
                {//ομάδα εντολών_2
                 //Αρνητικός αριθμός
                }

        else
                {//ομάδα εντολών_3
                 //θετικός αριθμός
                }
```

Υλοποίηση If-then-else-if (2)



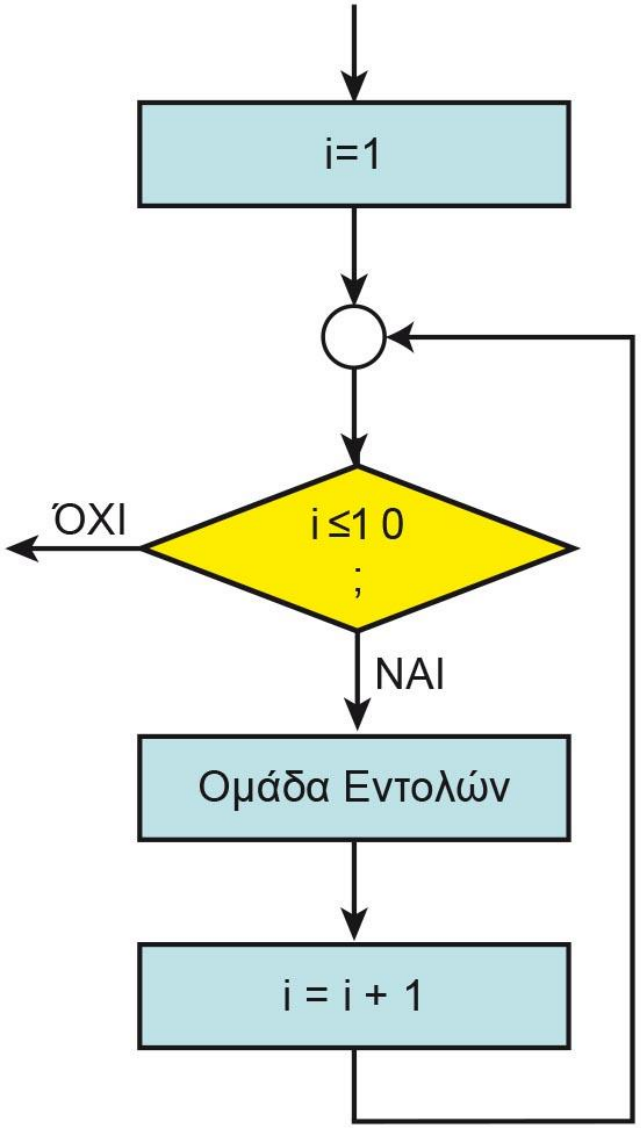
Δομή επανάληψης **do-while**



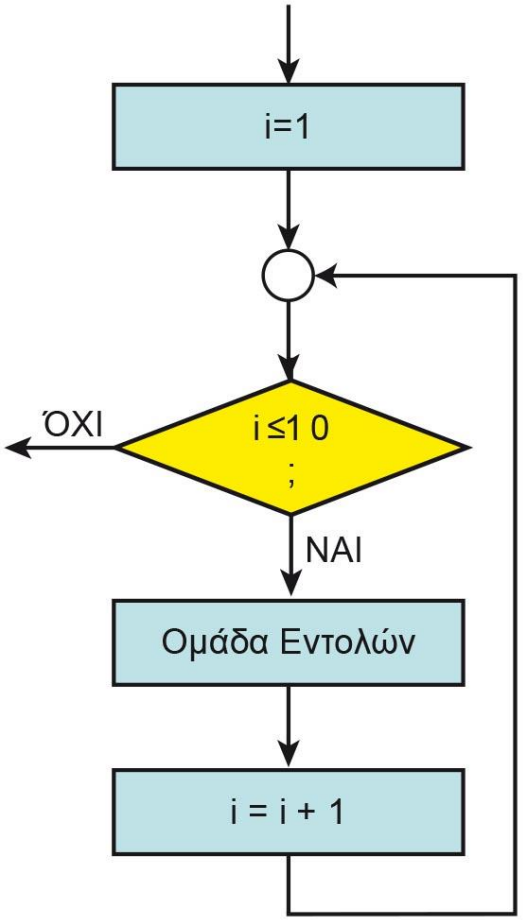

```
i=1
do
{

//Ομάδα εντολών

i=i+1;

}

while (i<=10)
```

```
MOV BH,1
start:


;Ομάδα εντολών

INC BH ; BH=BH+1


CMP BH,10
JLE start
```

Δομή επανάληψης **while-do**



Δομή επανάληψης **while-do**



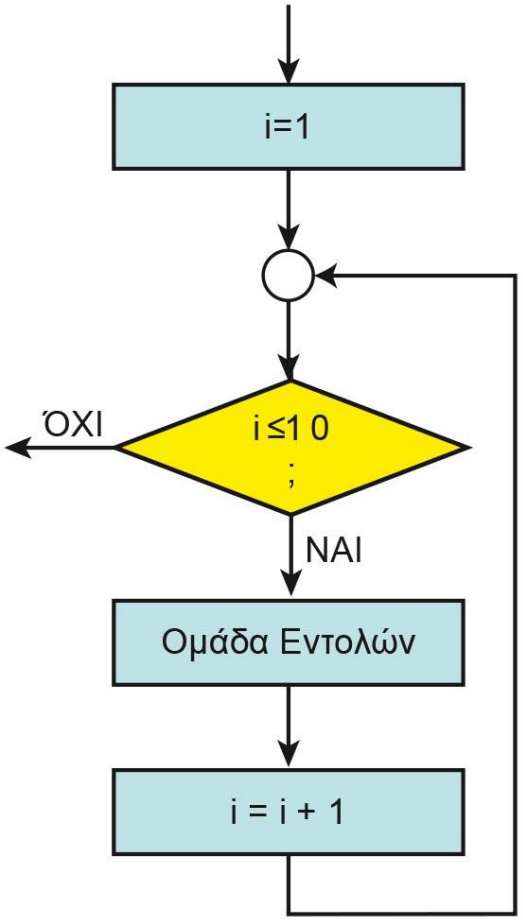

```
        MOV  BH,1
begin:
        CMP  BH, 10
        JLE  start
        JMP  exit
start:
        ; Ομάδα εντολών
        INC BH
        JMP begin
exit:
```

BH>10

BH≤10

Αρχικός έλεγχος
B<=10

Δομή επανάληψης **while-do**




```
        MOV  BH,1
begin:
        CMP  BH, 10
        JG   exit
start:
        ; Ομάδα εντολών
        INC BH
        JMP begin
exit:
```

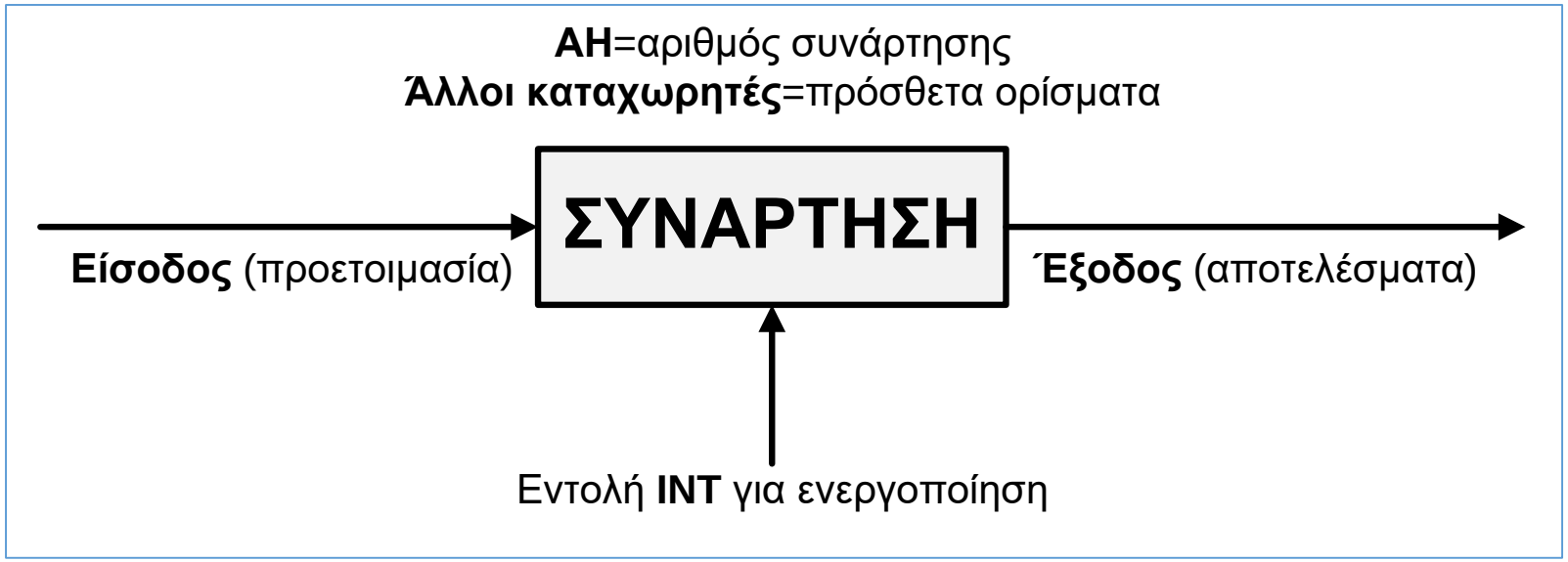
BH≤10

BH>10

Αρχικός έλεγχος
B>10

Πρακτική Εφαρμογή 1 : Απλή εμφάνιση μηνύματος

```
format MZ

push CS
pop DS

mov AH,9
mov DX,my_message
int 21h

mov AX,4C00h

int 21h

my_message db 'My first program!','$'
```



Τα προγράμματα στην πράξη

Δοκιμάστε τις δομές επανάληψης που έχουν ήδη παρουσιαστεί με την εμφάνιση μηνύματος

Δοκιμάστε την εντολή loop. Ποιες εντολές αντιπροσωπεύει;

Εμφάνιση μηνυμάτων με χρονική καθυστέρηση

Μερικές αριθμητικές εντολές (παραδείγματα)

Πρόσθεση

ADD AX,BX (AX=AX+BX)

ADD AX,[SI] (AX=AX+[SI])

ADD [DI],AL ([DI]=[DI]+AL)

ADD AL,09 (AL=AL+09)

ADD [ετικέτα],09 ([ετικέτα]=[ετικέτα]+09)

Αύξηση κατά 1

INC AX (AX=AX+1)

INC byte [BX] ([BX]=[BX]+1, 1 θέση)

Μείωση κατά 1

DEC AX (AX=AX-1)

DEC byte [BX] ([BX]=[BX]-1, 1 θέση)

Πολλαπλασιασμός

MUL BL (AX=AL*BL)

MUL BX (DX:AX=AX*BX)

Διαίρεση

DIV BX ([DX:AX]/BX, Π=AX, Υ=DX)

DIV BL (AX/BL, Π=AL, Υ=AH)

Εφαρμογή 2
Εμφάνιση γραφικής παράστασης

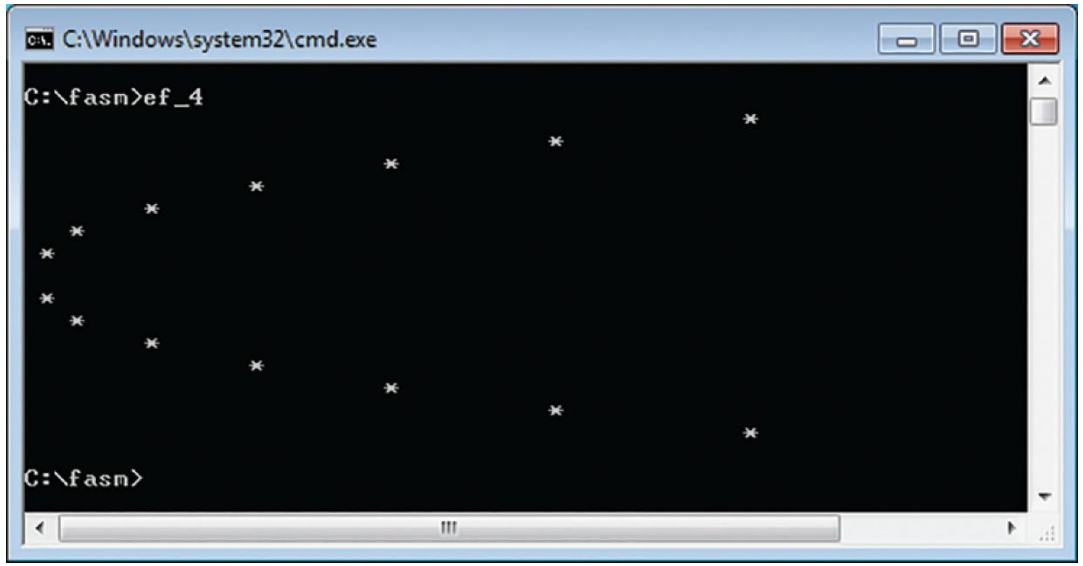
(α) Βρόχος επανάληψης για τις ακέραιες τιμές του X

(β) Υπολογισμός του $N=f(x)=X^2$ για όλες τις τιμές του X

(γ) Εμφάνιση N-1 κενών και ενός αστερίσκου ανά γραμμή

*** ΠΡΟΣΟΧΗ**

Η εμφάνιση γίνεται με αντεστραμμένους άξονες



```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Αρχικό Χ

Τελικό Χ

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

Υπολογισμός X^2
AX=AL*AL

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Αν $X^2=0$, απλά αλλάζουμε γραμμή

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Αν X^2<>0, υπολογίζουμε Y-1 (X^2-1)

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

Εμφάνιση κενών (Υ-1)

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

Εμφάνιση αστερίσκου

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Αλλαγή γραμμής

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Κλείσιμο βρόχου για το Χ

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

```
mov BL,-7

start:

mov AX,0000
mov AL,BL
mul AL
mov CL,AL

cmp CL,0
jz Line
dec CL

start2:
        mov AH,09
        mov DX,keno
        int 21h
        dec CL
        cmp CL,0
jg start2
```

Έξοδος

```
mov AH,09
mov DX,star
int 21h

Line:

mov AH,09
mov DX,nline
int 21h

inc BL
cmp BL,7

jle start

exit:

mov AX,4C00h
int 21h

star    db '*','$'
keno    db ' ','$'
nline   db 0Dh,0Ah,'$'
```

Εφαρμογή 3
Ανάγνωση μονοψήφιου αριθμού και έλεγχος εγκυρότητας

Σύμβολο	Κώδικας ASCII	Αφαίρεση	Καθαρός αριθμός
‘0’	30h	30h-30h=0	0
‘1’	31h	31h-30h=1	1
‘2’	32h	33h-30h=2	2
‘3’	33h	33h-30h=3	3
‘4’	34h	34h-30h=4	4
‘5’	35h	35h-30h=5	5
‘6’	36h	36h-30h=6	6
‘7’	37h	37h-30h=7	7
‘8’	38h	38h-30h=8	8
‘9’	39h	39h-30h=9	9

Εφαρμογή 3
Ανάγνωση μονοψήφιου αριθμού και έλεγχος εγκυρότητας

```
mov AH,09
mov DX,readmsg
int 21h
```

Εμφάνιση βοηθητικού μηνύματος

```
mov AH,1
int 21h
```

Διάβασμα ενός συμβόλου

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov CL, AL
sub CL, 30h

cmp CL, 0
jl error

cmp CL,9
jg error
```

Μετατροπή σε αριθμητική αξία

Εφαρμογή 3
Ανάγνωση μονοψήφιου αριθμού και έλεγχος εγκυρότητας

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

 mov CL, AL
 sub CL, 30h

 cmp CL, 0
 jl error

 cmp CL,9
 jg error
```

Έλεγχος αν ο αριθμός ανήκει στο διάστημα [0,9]

Αν δεν ανήκει, πηγαίνουμε στο error

Εφαρμογή 3
Ανάγνωση μονοψήφιου αριθμού και έλεγχος εγκυρότητας

Εμφάνιση μηνύματος ότι ανήκει

Εμφάνιση μηνύματος ότι ΔΕΝ ανήκει

```
 mov AH,09
   mov DX,yes_msg
   int 21h
   jmp exit

error:

      mov AH,09
      mov DX,no_msg
      int 21h
;=============
exit:
      mov   AX,4C00h
      int   21h
;==============
     readmsg  db 'Arithmos: ','$'
     yes_msg  db 0DH,0AH,'OK! Einai arithmos [0,9] ','$'
     no_msg  db 0DH,0AH,'Error: Den einai arithmos ','$'
```

Εφαρμογή 3
Ανάγνωση μονοψήφιου αριθμού και έλεγχος εγκυρότητας

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

 mov CL, AL
 sub CL, 30h

 cmp CL, 0
 jl error

 cmp CL,9
 jg error
```

```
 mov AH,09
   mov DX,yes_msg
   int 21h
   jmp exit

error:

      mov AH,09
      mov DX,no_msg
      int 21h
;=============
exit:
      mov   AX,4C00h
      int   21h
;==============
     readmsg  db 'Arithmos: ','$'
     yes_msg  db 0DH,0AH,'OK! Einai arithmos [0,9] ','$'
     no_msg  db 0DH,0AH,'Error: Den einai arithmos ','$'
```

Να γραφεί πρόγραμμα που να υπολογίζει το $y=ax^2+bx$

Αν y ανήκει στο διάστημα [1,10], τότε θα γίνεται υπολογισμός του αθροίσματος 1+…+5

Αν y=0, θα γίνεται ο υπολογισμός (x+1)*(x+2), διαφορετικά αν y>15, τότε θα εμφανίζεται ένα μήνυμα

Το x θα είναι μονοψήφιος αριθμός που θα εισάγεται από το πληκτρολόγιο

Να γραφεί πρόγραμμα που να υπολογίζει το $y=ax^2+bx$

Αν y ανήκει στο διάστημα [1,10], τότε θα γίνεται υπολογισμός του αθροίσματος 1+…+5

Αν y=0, θα γίνεται ο υπολογισμός (x+1)*(x+2), διαφορετικά αν y>15, τότε θα εμφανίζεται ένα μήνυμα

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

Παράδειγμα
Μετατροπή της ακολουθίας “15”
σε αριθμητική αξία

(31h-30h)*10+(35h-30h)*1 = 10+5 = 15

ή

(‘1’-30h)*10+(‘5’-30h)*1 = 10+5 = 15

```
mov AH,09
mov DX,readmsg
int 21h
```

Ανάγνωση 1ου συμβόλου

```
mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL
```

```
mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL


mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

Μετατροπή 1ου συμβόλου σε αριθμητική αξία

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL


mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

Υπολογισμός δεκάδων

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL


mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

Ανάγνωση 2ου συμβόλου

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL


mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

Μετατροπή 2ου συμβόλου σε αριθμητική αξία

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

Υπολογισμός τελικής αξίας αριθμού

Υποθετικός έλεγχος ορθότητας

Εμφάνιση μηνύματος αν DL=12

```
  add DL,CL

  cmp DL,12
  jne exit


  mov AH,09
  mov DX,ok_msg
  int 21h

;============
exit:

     mov   AX,4C00h
     int   21h

;=========
  ok_msg  db 0Ah, 0Dh, 'Arithmos=12 ','$'
  readmsg db ':','$'
```

Εφαρμογή 4
Ανάγνωση διψήφιου αριθμού

```
mov AH,09
mov DX,readmsg
int 21h


mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL


mov AH,1
int 21h

mov CL, AL
sub CL, 30h
```

```
  add DL,CL

  cmp DL,12
  jne exit


  mov AH,09
  mov DX,ok_msg
  int 21h

;============
exit:

     mov   AX,4C00h
     int   21h

;=========
  ok_msg  db 0Ah, 0Dh, 'Arithmos=12 ','$'
  readmsg db ':','$'
```

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

Υπολογισμός $1^2+2^2+3^2 = 14$

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

Αρχικοποίηση

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Υπολογισμός τετραγώνου

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Υπολογισμός αθροίσματος

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Κλείσιμο βρόχου

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Υποθετικός έλεγχος ορθότητας

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Εμφάνιση μηνύματος αν CX=14

Εφαρμογή 5
Υπολογισμός αθροίσματος τετραγώνων

```
            mov AX,0
            mov CX,0
            mov BL,1
start:
            mov AL,BL
            mul AL

            add CX,AX
            inc BL

            cmp BL,3
jle start

            cmp CX,14
            jne exit

            mov AH,09
            mov DX,msg
            int 21H
 exit:
 mov   AX,4C00h
 int   21h

msg    db  'S=14','$'
```

Εφαρμογή 6
Υπολογισμός αθροίσματος παραμετρικά

Υπολογισμός
$1^2+2^2+3^2+\ldots+(N-2)^2+(N-1)^2+N^2$

(α) Πολλαπλασιασμός
με 8bit όρισμα

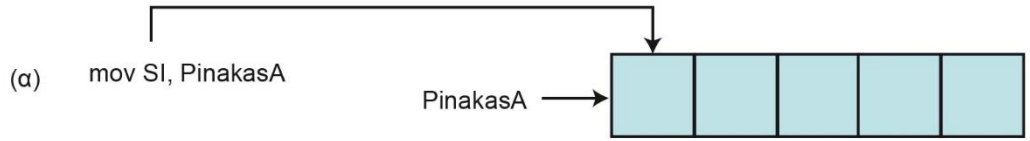
Εφαρμογή 6

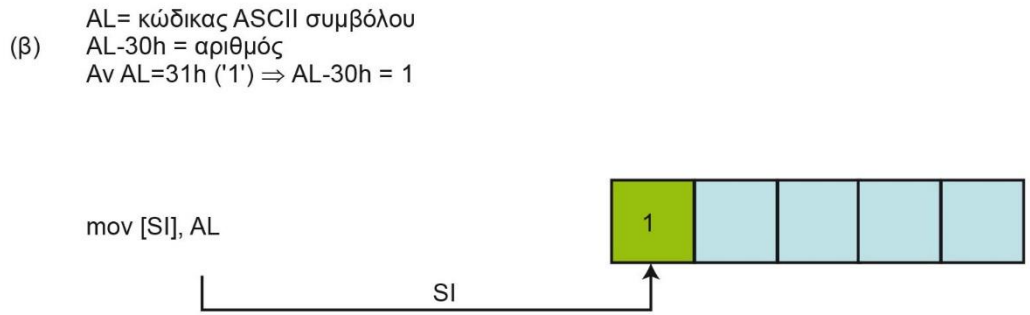
```
mov AH,09
mov DX,readmsg
int 21h

mov AH,1
int 21h

mov DL, AL
sub DL, 30h

mov AX,000ah

mul  DL
mov DL,AL

mov AH,1
int 21h

mov CL, AL
sub CL, 30h
add DL,CL
mov AX,0
mov CX,0
mov BL,1
start:
        mov AL,BL
        mul AL

        add CX,AX
        inc BL
        cmp BL,DL

jle start
        cmp CX,506
        jne exit

        mov AH,09
        mov DX,msg
        int 21H

 exit:
mov   AX,4C00h
int   21h

readmsg db 'N:','$'
msg     db     0Ah, 0Dh, 'S=506','$'
```

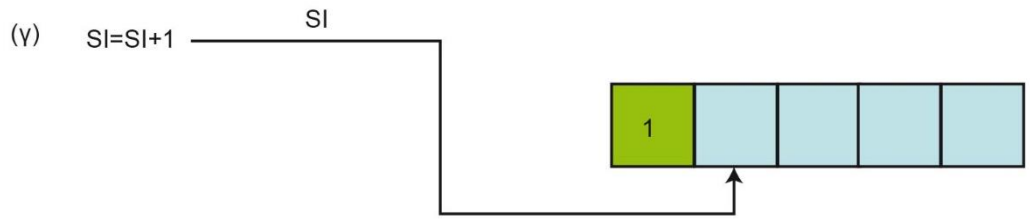
(α) Να αναλυθεί η λειτουργία του προγράμματος

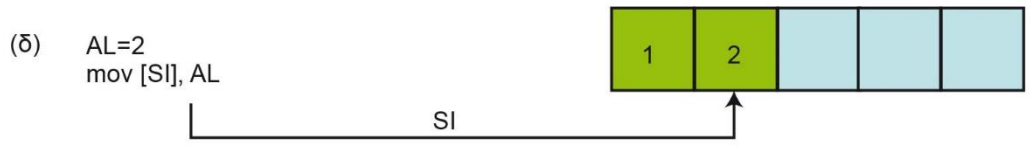
(β) Ποιον υπολογισμό πραγματοποιεί ;

(γ) Με πόσα bit γίνεται ο υπολογισμός ;





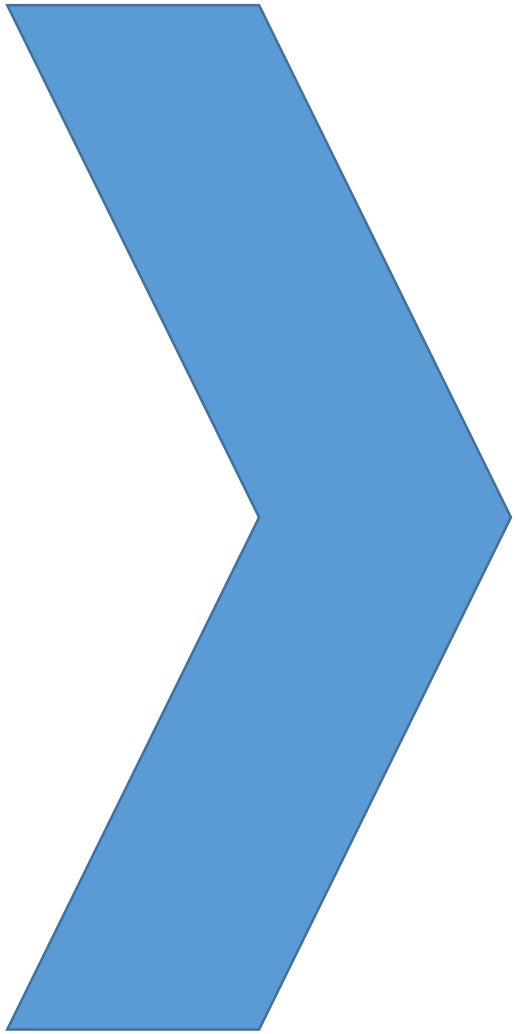




Εφαρμογή 7
Βασική διαχείριση μονοδιάστατου πίνακα

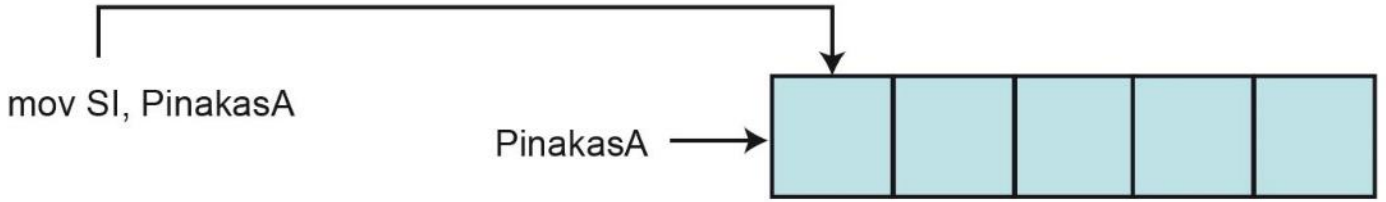
- Μετατροπή συμβόλου σε αριθμό
- Αποθήκευση στον πίνακα

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```



Γέμισμα πίνακα

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```



Το SI δείχνει τη διεύθυνση PinakasA

Μέσω του SI έχουμε πρόσβαση στον πίνακα

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Εμφάνιση βοηθητικού μηνύματος για την εισαγωγή των στοιχείων

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Ανάγνωση συμβόλου από το πληκτρολόγιο

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Μετατροπή συμβόλου σε αριθμό

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Μετατροπή συμβόλου σε αριθμό

Αποθήκευση αριθμού στη διεύθυνση που δείχνει το SI

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Μετατροπή συμβόλου σε αριθμό

Αποθήκευση αριθμού στη διεύθυνση που δείχνει το SI

Το SI θα δείχνει τώρα στην επόμενη διεύθυνση

```
format MZ
  push  CS
  pop   DS
;====================
  mov SI, pinakasA

  mov BL,1
  again:

            mov AH,09
            mov DX,msg
            int 21h

            mov AH,1
            int 21h

            sub AL,30h
            mov [SI],AL
            inc SI

            inc BL
            cmp BL,5
  jle again
```

Αύξηση μετρητή βρόχου

Έλεγχος οριακής τιμής

```
mov AH,09
mov DX,nline
int 21h

   mov SI, pinakasA
   mov BL,1
again2:
            mov AX,0002
            mov CL,[SI]
            mul CL
            mov [SI],AL
            inc SI
            inc BL
            cmp BL,5
  jle again2
```

Ποιος υπολογισμός γίνεται ;

Εξηγήστε τη λειτουργία της ενότητας

```
 mov SI, pinakasA
 mov DI, Xout

 mov BL,1
again3:
        mov AL,[SI]
        add AL,30h
        mov [DI],AL

        mov AH,09
        mov DX,Xout
        int 21h

        inc SI
        inc BL
        cmp BL,5
jle again3
```

Με ποιο τρόπο γίνεται η εμφάνιση του περιεχομένου του πίνακα στην οθόνη ;

```
 mov SI, pinakasA
 mov DI, Xout

 mov BL,1
again3:
        mov AL,[SI]
        add AL,30h
        mov [DI],AL

        mov AH,09
        mov DX,Xout
        int 21h

        inc SI
        inc BL
        cmp BL,5
jle again3
```

Έστω ο πίνακας:

pinakasA [5, 6, 2, 1, 0]

Στην πρώτη εκτέλεση του βρόχου

mov AL,[SI] ; **AL=5**
Add AL,30h ; **AL=35h='5'**
mov [DI],AL ; **Προετοιμασία μηνύματος**

```
;=====================
 exit:

      mov   AX,4C00h
      int   21h

;====================
pinakasA              db ?, ?, ?, ?, ?
Xout       db 'X',0Ah,0Dh,'$'
msg                   db ':','$'
nline      db 0Ah,0Dh,'$'
```

ΔΡΑΣΤΗΡΙΟΤΗΤΕΣ

- Μπορείτε να χρησιμοποιήσετε τον DOS assembler από τη διεύθυνση https://flatassembler.net/, προκειμένου να δοκιμάστε τα προγράμματα INTEL 16bit που παρουσιάστηκαν προηγουμένως.

- Στη διεύθυνση https://flatassembler.net/ , μπορείτε να βρείτε έναν προσομοιωτή για τον μικροεπεξεργαστή MIPS, για να δοκιμάσετε το πρόγραμμα εμφάνισης του μηνύματος «HELLO» από το βιβλίο.

- Να συγκρίνετε τον προγραμματισμό των μικροεπεξεργαστών 6502, Z80, 8086 και MIPS R2000 με επίκεντρο την εμφάνιση του μηνύματος «HELLO». Σχολιάστε αναλυτικά ομοιότητες και διαφορές.