v.25-03-2024


# ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
# ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
# ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

## ΜΑΘΗΜΑ
## ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ (206ΕΥΥΚ)
## ΠΠΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
## ΕΑΡΙΝΟ 2023-2024

## Διάλεξη Νº3:
### M68000: Εντολές (MOVE, ADD, CMP, Bcc, Logic, Shift, Bit)

**Δ. Καραμπατζάκης, Επίκουρος Καθηγητής**

**email. dkara@cs.ihu.gr**

# Δήλωση προσβασιμότητας

Σε αυτό το μάθημα όλες/οι οι φοιτήτριες/τές απολαμβάνουν – και αντίστοιχα υποχρεούνται να σέβονται – το δικαίωμα της ίσης μεταχείρισης. Δεν είναι ανεκτή και αποδεκτή κανενός τύπου και μορφής διάκριση με κριτήρια την εθνικότητα, τη φυλή, την καταγωγή, τη γλώσσα, το φύλο, τη θρησκεία, την ηλικία, την υγεία, τη σωματική ικανότητα, την ιδιωτική ζωή, τον γενετήσιο προσανατολισμό, τη σωματική ικανότητα και την οικονομική και κοινωνική κατάσταση στην οποία αυτοί βρίσκονται.

Το Πανεπιστήμιο άγρυπνα μεριμνά για τη διασφάλιση της αρχής των ίσων ευκαιριών και της ίσης μεταχείρισης. Οι κοινωνικές προκαταλήψεις και οι ιδεολογικές παρωπίδες είναι έννοιες τελείως ξένες με την επιστημονική πρόοδο την οποία το Πανεπιστήμιο είναι ταγμένο να υπηρετεί.

Ο Διδάσκων

# Πληροφορίες για το Μάθημα

**Διδάσκων:**

Δημήτρης Καραμπατζάκης, Επίκουρος Καθηγητής

Αναλογικά και Ψηφιακά Ηλεκτρονικά Συστήματα

Μέλος Εργαστηρίου Βιομηχανικών και Εκπαιδευτικών Ενσωματωμένων Συστημάτων

**Επικοινωνία / πληροφορίες:**

Email. dkara@cs.ihu.gr

web. http://www.internetofthings.gr/

**Ώρες Γραφείου:**

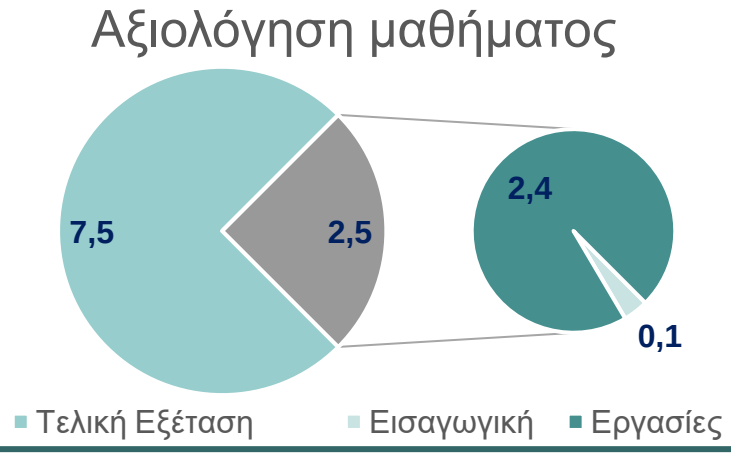
μετά από συνεννόηση με email στο ΦΕ 315 (πάνω από αιθ. Α1)

# Πληροφορίες για το Μάθημα (Γενικές)

- **Κάθε Τρίτη, Πέμπτη 12.00 π.μ. - 14.00 μ.μ. μάθημα θεωρίας στο Μεγάλο Αμφιθέατρο (μπορεί να αλλάζει με ανακοινώσεις).**
- **Η διαχείριση του μαθήματος θα γίνει με χρήση της υπηρεσίας https://courses.cs.ihu.gr**
- **Όλοι οι φοιτητές πρέπει να έχουν λογαριασμό στο uregister.**
- **Η ιστοσελίδα με τις πληροφορίες του μαθήματος: http://iees.cs.ihu.gr/?page_id=3209**
- **Υλικό του μαθήματος στο moodle: https://moodle.cs.ihu.gr/**

# Πληροφορίες για το Μάθημα (Αξιολόγηση)

- **Η βαθμολογία είναι 75% από την τελική εξέταση και 25% από τις ατομικές εργασίες (1 σετ ασκήσεων) που θα δοθούν για το σπίτι.**
- **Η τελική εξέταση είναι με ανοιχτό το κύριο σύγγραμμα του μαθήματος.**
- **Ο βαθμός του μαθήματος (ΒΜ = ΓΕ*0,75 + ΣΑ*0,25) πρέπει να είναι τουλάχιστον πέντε (5).**



# Πληροφορίες για το Μάθημα (Μονάδες)

- **Κωδικός Μαθήματος:** 206ΕΥΥΚ
- **Εξάμηνο:** 2ο
- **Τύπος Μαθήματος:** Υποβάθρου, Ανάπτυξης Δεξιοτήτων
- **Είδος Μαθήματος:** Υποχρεωτικό (ΥΠ)
- **Διδασκαλία Θεωρίας:** 3 ώρες/εβδομάδα
- **Διδασκαλία Φροντιστήριο:** 1 ώρες/εβδομάδα
- **Πιστωτικές μονάδες ECTS: 7**
- **Γλώσσα διδασκαλίας και Εξετάσεων:** Ελληνικά

# Πληροφορίες για το Μάθημα (Φόρτος)

| | |
|---|---|
| **Δραστηριότητα** | **Φόρτος εργασίας εξαμήνου** |
| **Διαλέξεις** | 78 ώρες |
| **Φροντιστηριακές Ασκήσεις** | 26 ώρες |
| **Γραπτές Εξετάσεις** | 2 ώρες |
| **Γραπτές Εργασίες** | 34 ώρες |
| **Αυτοτελής Μελέτη** | 35 ώρες |
| **Σύνολο** | **175 ώρες (7 ECTS)** |

# Κύριο Σύγγραμμα Μαθήματος (ΕΥΔΟΞΟΣ)


**Οργάνωση και Σχεδίαση Υπολογιστών**

Συγγραφέας: Πογαρίδης Δημήτριος

Έτος Έκδοσης: 2019

Κωδικός στον Εύδοξο: **86192986**

# Λογισμικό - Αναπτυξιακό

- **Α’ μέρος μαθήματος (CISC):**
  - Assembly για τον Motorola68000
  - Λογισμικό easy68k http://www.easy68k.com/
- **Β’ μέρος μαθήματος (RISC):**
  - Υλοποίηση σχεδιάσεων σε αναπτυξιακό Arduino (προαιρετική αγορά του υλικού σύμφωνα με τις οδηγίες)
  - Λογισμικό Arduino IDE https://www.arduino.cc/en/Main/Software
  - Η γλώσσα προγραμματισμού (C++) και οι εντολές που υποστηρίζει είναι διαθέσιμες στο: https://www.arduino.cc/reference/en/

# Οι εντολές του M68000

# Οι εντολές του M68000

**Οι εντολές του M68000 διαιρούνται στις παρακάτω ομάδες:**

1. *Εντολές μεταφοράς δεδομένων*
2. *Αριθμητικές εντολές*
3. *Αριθμητικές εντολές ακεραίων αριθμών*
4. *Αριθμητικές εντολές δεκαδικών αριθμών*
5. *Λογικές εντολές*
6. *Εντολές ολίσθησης και περιστροφής*
7. *Εντολές σύγκρισης και ελέγχου*
8. *Εντολές διακλάδωσης*
9. *Εντολές διαχείρισης υπορουτινών*
10. *Εντολές διαχείρισης ψηφίων*

# Εντολές Μεταφοράς Δεδομένων

# MOVE.W D0,D7

**Πριν την εκτέλεση της εντολής**

**[D0]=$12345678**

**[D7]=$87654321**

**Μετά την εκτέλεση της εντολής**

**[D0]=$12345678**

**[D7]=$87655678**

**Οι Δείκτες του Καταχωρητή Κατάστασης**
**X=0, N=0, Z=0, V=0, C=0**

# MOVEA.L (A0),A2

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400500, [$00400500]=$00403000, [A2]=$00400600**

**Μετά την εκτέλεση της εντολής**

**[A2]= $00403000**

MOVE ADDRESS

Syntax: MOVEA <ea>,An  Size = word, longword

* If it is a word, it is sign-extended to a longword

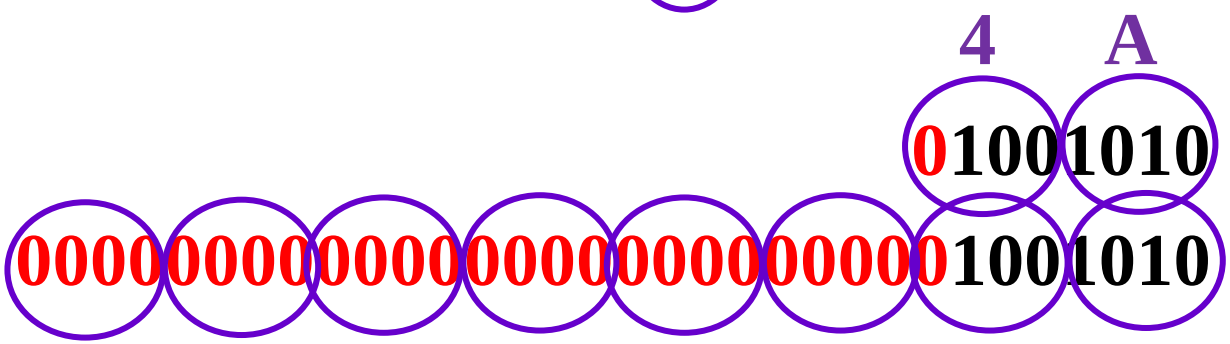
** The conditions codes are not affected

# MOVEQ #$4A,D0

**Πριν την εκτέλεση της εντολής**

**[D0]=$HHHHHHHH**

**Μετά την εκτέλεση της εντολής**
**[D0]=$0000004A**




MOVE quick a literal

Syntax: MOVEQ #<data>,Dn  Size = longword

* The literal data is 8-bit, -128 to +127

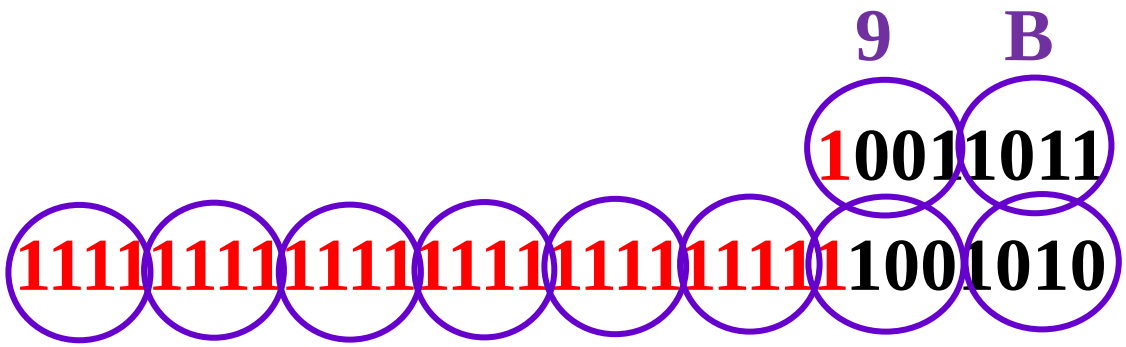
** when it is transferred, it is sign-extended to a longword

# MOVEQ #$9B,D0

**Πριν την εκτέλεση της εντολής**

**[D0]=$HHHHHHHH**

**Μετά την εκτέλεση της εντολής**
**[D0]=$FFFFFF9B**



# MOVEM

**Syntax**

**MOVEM.s <ea>,<register list>**

**MOVEM.s <register list>,<ea>**

**Size = word, longword**

**<register list>:**

1. **Rn – a single register**
2. **Rn-Rm – a range of registers (m>n)**
3. **Any combination of 1. and 2. separated by a slash /**

**Examples**

**MOVEM.L D0-D7/A0-A6,$1234**

**MOVEM.L (A5),D0-D2/D5-D7/A0-A3/A6**

**MOVEM.W (A7)+,D0-D5/D7/A0-A6**

**MOVEM.W D0-D5/D7/A0-A6,-(A7)**

**A. a group of register is transferred to OR from Memory**
**(using AddrMode different than –(An) / (An)+)**
**MOVEM.L D0-D2/D4/A5/A6,$1234**
**Moves Regs D0,D1,D2,D4,A5,A6 to Memory, starting at location $1234 (D0) and moving to $1238**
**(.L +4 bytes, .W +2 bytes)**
**The order of transfer is D0 to D7, A0 to A7**

**B. If <ea> is –(An) only Reg to Mem operation is permitted**
**MOVEM.W D0-D5/D7/A0-A6,-(A7)**
**Moves Regs A6-A0,D7,D5-D0 to Memory,**
**starting at location (A7 minus .L 4 bytes, .W 2 bytes) and down through lower addresses (Multiple PUSH)**
**The order of transfer is A7 to A0, D7 to D0**

**C. If <ea> is (An)+ only Mem to Reg operation is permitted**

**MOVEM.W (A7)+,D0-D5/D7/A0-A6**

**Moves data from Mem to Regs D0-D5/D7/A0-A6, starting at Mem (A7 plus .L 4 bytes, .W 2 bytes) and up through higher addresses (Multiple POP)**

**The order of transfer is D0 to D7, A0 to A7**

**TIP: MOVEM.W sign-extends words when they are moved to Data Regs.**

# MOVEM: Instruction

Instruction: MOVEM.W D0-D3,(A0)

Register Contents

Before:

| | |
|---|---|
| D0 | 55556666 |
| D1 | 77778888 |
| D2 | 9999AAAA |
| D3 | BBBBCCCC |
| A0 | 000030B8 |

*****Memory*****

After :

| Address | Content |
|---|---|
| 0030B8 | 66 66 |
| 0030BA | 88 88 |
| 0030BC | AA AA |
| 0030BE | CC CC |

Instruction: MOVEM.L D0-D3,(A0)

Register Contents

Before:

| | |
|---|---|
| D0 | 55556666 |
| D1 | 77778888 |
| D2 | 9999AAAA |
| D3 | BBBBCCCC |
| A0 | 000030B8 |

*****Memory*****

After :

| Address | Content |
|---|---|
| 0030B8 | 55 55 |
| 0030BA | 66 66 |
| 0030BC | 77 77 |
| 0030BE | 88 88 |
| 0030C0 | 99 99 |
| 0030C2 | AA AA |
| 0030C4 | BB BB |
| 0030C6 | CC CC |

# MOVEM: Instruction

```
Instruction: MOVEM.L (A0),D0-D3

  Before: *****Memory*****
          Address Content
           0030B8  11 11
           0030BA  AA AA
           0030BC  22 22
           0030BE  BB BB
           0030C0  33 33
           0030C2  CC CC
           0030C4  44 44
           0030C6  DD DD
          A0   000030B8

  After : Register Contents
          D0   1111AAAA
          D1   2222BBBB
          D2   3333CCCC
          D3   4444DDDD
```

```
Instruction: MOVEM.W (A0),D0-D3

  Before: *****Memory*****
          Address Content
           0030B8  11 11
           0030BA  AA AA
           0030BC  22 22
           0030BE  BB BB
           0030C0  33 33
           0030C2  CC CC
           0030C4  44 44
           0030C6  DD DD
          A0   000030B8

  After : Register Contents
          D0   00001111
          D1   FFFFAAAA
          D2   00002222
          D3   FFFFBBBB
```

Sign-extends

# MOVEM.W $400500,D0/D5
# MOVEM.W D0/D5,$400600

**Πριν την εκτέλεση των εντολών**

**[$400500-1]=$3FFE, [$400502-3]=$F40B**

**[D0]=$HHHHHHHH, [D5]=$HHHHHHHH**

**Μετά την εκτέλεση της πρώτης εντολής**

**[D0]=$00003FFE, [D5]= $FFFFF40B**

**Μετά την εκτέλεση της δεύτερης εντολής**

**[$400600-1]=$3FFE, [$400602-3]=$F40B**

# MOVEP.L D0,0(A0)

**Πριν την εκτέλεση των εντολών**

**[D0]=$12345678, [A0]=$00400600**

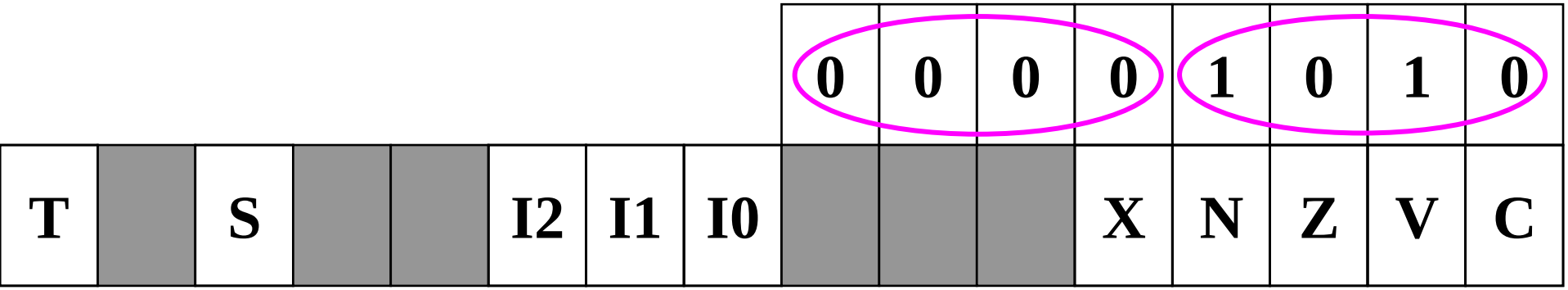
**Μετά την εκτέλεση της εντολής**

**[$400600]=$12, [$00400602]=$34,**
**[$400604]=$56, [$00400606]=$78**

# MOVE #$000A,CCR

**Πρόκειται για εντολή μήκους λέξης που όμως επηρεάζει μόνο το byte του κώδικα συνθήκης του καταχωρητή κατάστασης**

**Μετά την εκτέλεση της εντολής**
**[CCR]=$0A**




**Οι Δείκτες του Καταχωρητή Κατάστασης**
**X=0, N=1, Z=0, V=1, C=0**

# MOVE D2,SR και MOVE SR,D5

## (Προνομιούχες Εντολές ή Priveledge Instructions)
## Εκτελούνται μόνο σε κατάσταση επόπτη

**Πριν την εκτέλεση των εντολών**

**[D2]=$00000011**

**[D5]=$0000F351**

**Μετά την εκτέλεση της πρώτης εντολής**
**[SR]=$0011**

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| T | | S | | | I2 | I1 | I0 | | | | X | N | Z | V | C |

**Δείκτες Καταχωρητή Κατάστασης X=1, N=0, Z=0, V=0, C=1**

**Μετά την εκτέλεση της δεύτερης εντολής [D5]=$00000011**

# MOVE USP,A0 και MOVE A1,USP

## (Προνομιούχες Εντολές ή Priveledge Instructions)
## Εκτελούνται μόνο σε κατάσταση επόπτη

**Πριν την εκτέλεση των εντολών**

**[A0]=$00400300, [A1]=$00400400, [USP]=$00405000**

**Μετά την εκτέλεση της πρώτης εντολής**
**[A0]=$00405000**

**Μετά την εκτέλεση της δεύτερης εντολής**
**[USP]=$00400400**

# LEA 4(A3,D1),A0

**Πριν την εκτέλεση της εντολής**

**[A3]=$00400500, [D1]=$000010C3**

**Μετά την εκτέλεση της εντολής**

**[A0]=$004015C7**

**[A0]=[A3]+[D1]+4 = $00400500+$000010C3+4=$004015C7**

LOAD EFFECTIVE ADDRESS

Syntax: LEA <ea>,An Size = longword

**MOVEA.L #$1234,A0 = = LEA $1234,A0**

**MOVEA.W #$8000,A0 -> $FFFF8000 SIGN-EXTENSION**

**LEA $8000,A0 -> $0008000 NO SIGN-EXTENSION**

# EXG A3,D1

**Πριν την εκτέλεση της εντολής**

**[A3]=$0040351A, [D1]=$0040067D**

**Μετά την εκτέλεση της εντολής**

**[A3]=$0040067D, [D1]=$0040351A**

EXCHANGE REGISTERS

Syntax: EXG $R_x$ , $R_y$ Size = longword

# SWAP D5

**Πριν την εκτέλεση της εντολής**

**[D5]=$0000FFFF**

**Μετά την εκτέλεση της εντολής**

**[D5]=$FFFF0000**

SWAP DATA REGISTERS HALVES

Syntax: SWAP $D_n$ Size = word

# CLR D5

**Πριν την εκτέλεση της εντολής**

**[D5]=$9B62F4C1**

**Μετά την εκτέλεση της εντολής**

**[D5]=$00000000**

CLEAR AN OPERAND

Syntax: CLR <EA>  Size = byte, word, longword

CLR δεν μπορεί να χρησιμοποιηθεί στους καταχωρητές διευθύνσεων

# Αριθμητικές Εντολές

# Μη προσημασμένη-Προσημασμένη Αριθμητική Απλής και Πολλαπλής Ακρίβειας

**Στη μη-προσημασμένη αριθμητική ένα byte παίρνει τιμές από 0-$255_{10}$, μια λέξη τιμές από 0-$65.535_{10}$ και μια μακριά λέξη 0-$4.294.967.295_{10}$.**

**Στην προσημασμένη αριθμητική το περισσότερο σημαντικό ψηφίο της ψηφιολέξης παριστάνει το πρόσημο του αριθμού.**

**Όταν το ψηφίο αυτό είναι "0" ο αριθμός είναι θετικός, ενώ όταν το ψηφίο αυτό είναι "1" ο αριθμός είναι αρνητικός.**

**Επομένως στη προσημασμένη αριθμητική:**

**Ένα byte μπορεί να πάρει τις τιμές από:**

**$(-128_{10})$ έως $(+127_{10})$**

**Μια λέξη τιμές από:**

**$(-32.768_{10})$ έως (+32.767)**

**Μια μακριά λέξη από:**

**$(-2.147.483.648_{10})$ έως $(2.147.483.647_{10})$**

**Για τον υπολογιστή όλοι οι αριθμοί θεωρούνται μη προσημασμένοι και κάνει τις πράξεις με τον ίδιο ακριβώς τρόπο.**

**Είναι ο χρήστης που καθορίζει αν πρόκειται για προσημασμένη ή μη-προσημασμένη αριθμητική.**

**Αν πρόκειται για μη-προσημασμένη αριθμητική τα δεδομένα και το αποτέλεσμα ερμηνεύονται ως μη-προσημασμένοι αριθμοί.**

**Αν πρόκειται για προσημασμένη αριθμητική τα δεδομένα και το αποτέλεσμα ερμηνεύονται ως προσημασμένοι αριθμοί.**

**Στη μη-προσημασμένη αριθμητική την υπερχείλιση τη δείχνει ο δείκτης κρατούμενου C.**

**Στην προσημασμένη αριθμητική την υπερχείλιση τη δείχνει ο δείκτης υπερχείλισης V.**

**Σύμφωνα με έναν κανόνα «υπερχείλιση υπάρχει όταν προστίθενται δύο ομόσημοι αριθμοί και το άθροισμά τους έχει αντίθετο πρόσημο»**

**Οι κανόνες ανίχνευσης υπερχείλισης (overflow) στην πρόσθεση με συμπλήρωμα ως 2 είναι απλές:**

1. **Αν το άθροισμα δύο θετικών αριθμών δώσει αρνητικό αποτέλεσμα,** τότε έχουμε υπερχείλιση.

2. **Αν το άθροισμα δύο αρνητικών αριθμών δώσει θετικό αποτέλεσμα**, τότε έχουμε υπερχείλιση.

3. Σε κάθε άλλη περίπτωση δεν έχουμε υπερχείλιση.

Είναι σημαντικό να αναφέρουμε ότι **υπερχείλιση (overflow)** και **κρατούμενο εξόδου (carry out)** μπορούν να προκύψουν το καθένα ξεχωριστά. Στους μη προσημασμένους αριθμούς, το κρατούμενο εξόδου είναι ίδιο με την υπερχείλιση.

Στους συμπλήρωμα ως προς 2, το κρατούμενο εξόδου δε μας λέει κάτι για την υπερχείλιση. Ο λόγος για τους παραπάνω κανόνες είναι γιατί η υπερχείλιση στους συμπλήρωμα ως προς 2 εμφανίζεται όχι όταν ένα bit βγαίνει από την αριστερή πλευρά, αλλά όταν ένα bit εισέρχεται στη θέση πρόσημου. Οι κανόνες ελέγχουν αυτή την μεταβολή με το πρόσημο του αποτελέσματος. Άρα αν έχουμε ομόσημους αριθμούς και προκύψει αποτέλεσμα με διαφορετικό πρόσημο τότε στο MSB έχουμε αλλαγή bit (από 0 σε 1 ή από 1 σε 0) και έχουμε υπερχείλιση.
Όταν ένας θετικός και ένας αρνητικός προστίθενται δεν μπορούν να υπερχειλίσουν γιατί το άθροισμά είναι μέσα στα όρια των προσθετέων. Όσο οι δύο προσθετέοι χωρούν εντός του επιτρεπόμενου εύρους αριθμών, τότε και το άθροισμά τους είναι μέσα σε αυτό το εύρος.

# Έστω ο δυαδικός αριθμός: $11000100_2 = C4_{16}$

**Ο αριθμός αυτός όταν ερμηνεύεται ως μη προσημασμένος αριθμός είναι ο δεκαδικός:**

$$11000100_2 = C4_{16} = (128+64+4) = 196_{10}$$

**Όταν όμως ερμηνεύεται ως προσημασμένος αριθμός τότε το περισσότερο σημαντικό ψηφίο δείχνει το πρόσημο.**

**Αν το ψηφίο αυτό είναι ¨0¨ τότε ο αριθμός είναι θετικός και το μέγεθός του δίνεται από τα υπόλοιπα εφτά ψηφία.**

**Αν το ψηφίο αυτό είναι ¨1¨ τότε ο αριθμός είναι αρνητικός και το μέγεθός του δίνεται από το συμπλήρωμα ως προς 2 των υπόλοιπων εφτά ψηφίων.**

**Στην προκειμένη περίπτωση ($11000100_2$) το περισσότερο σημαντικό ψηφίο είναι ¨1¨ και επομένως, ο αριθμός είναι αρνητικός με μέγεθος το συμπλήρωμα ως προς 2 των υπόλοιπων εφτά ψηφίων. Δηλαδή,**

**(Συμπλήρωμα ως προς 1) 0111011 +1=0111100=-60**

# Πρόσθεση δυαδικών αριθμών

Η πρόσθεση δυαδικών είναι παρόμοια με αυτή των δεκαδικών.

Για παράδειγμα έχουμε 8+7=15, που είναι μεγαλύτερο από το 9 (σε κάθε θέση χωρά ένα δεκαδικό) άρα κρατάμε το 5 και το 1 μεταφέρεται ως κρατούμενο.

Στην πρόσθεση δυαδικών αριθμών έχουμε ένα δυαδικό ψηφίο σε κάθε θέση.

Επομένως αν έχουμε $1+1 = 2_{10} = 10_2$ κρατάμε $0_2$ και το $1_2$ μεταφέρεται ως κρατούμενο.

Αν έχουμε $1+1+1 = 3_{10} = 11_2$ κρατάμε $1_2$ και το $1_2$ μεταφέρεται ως κρατούμενο.

# Πρόσθεση Απλής Ακρίβειας

**Όταν η πρόσθεση δύο αριθμών μήκους byte πρόκειται να δώσει άθροισμα μικρότερο από $255_{10}$, τότε χρησιμοποιείται πρόσθεση απλής ακρίβειας.**

***Περίπτωση 1η. Πρόσθεση δύο αριθμών με κρατούμενο. Αποτέλεσμα χωρίς κρατούμενο.***

| | | |
|---|---|---|
| | **00001101** | **13** |
| | **+11010011** | **+211** |
| **Κρατούμενο: C=0** | **$1110\ 0000_2$** | **$224_{10}$** |

## *Περίπτωση 2η. Πρόσθεση δύο αριθμών με κρατούμενο. Αποτέλεσμα με κρατούμενο.*

| | | |
|---|---|---|
| | **11111110** | **254** |
| | **+00000110** | **+6** |
| **Κρατούμενο: C=1** | **$00000100_2$** | **$4_{10}$** |

**Ο δείκτης κρατούμενου δίνει στον χρήστη να καταλάβει ότι το αποτέλεσμα έχει υπερβεί το 255 και γι' αυτό μπορεί να παρασταθεί ως $256+4 = 260_{10}$.**

# Πρόσθεση Πολλαπλής Ακρίβειας

**Η πρόσθεση πολλαπλής ακρίβειας χρησιμοποιείται όταν το αναμενόμενο αποτέλεσμα απ' την πρόσθεση δύο αριθμών μήκους byte είναι μεγαλύτερο του 255 ή όταν ο ένας τουλάχιστον απ' τους δύο αριθμούς είναι μεγαλύτερος του 255.**

**Στις περιπτώσεις αυτές οι προς πρόσθεση αριθμοί θα παρασταθούν με μια ψηφιολέξη δύο byte (16 ψηφίων) και η πρόσθεση θα γίνει με τον παρακάτω τρόπο.**

***Περίπτωση 1η. Πρόσθεση δύο αριθμών των 16 ψηφίων (δύο byte ο καθένας), χωρίς κρατούμενο απ' την πρόσθεση των λιγότερο σημαντικών byte.***

**$00000001\ 00000010_2 = 258_{10}$**

**$00010000\ 00010000_2 = 4112_{10}$**

**Πρόσθεση των λιγότερο σημαντικών bytes (C = 0)**

| | | |
|---|---|---|
| | **00000010** | **2** |
| | **+00010000** | **+16** |
| **Κρατούμενο: C=0** | **$00010010_2$** | **$18_{10}$** |

***Περίπτωση 1η. Πρόσθεση δύο αριθμών των 16 ψηφίων (δύο byte ο καθένας), χωρίς κρατούμενο απ' την πρόσθεση των λιγότερο σημαντικών byte.***

**$00000001\ 00000010_2 = 258_{10}$**

**$00010000\ 00010000_2 = 4112_{10}$**

**Πρόσθεση των περισσότερο σημαντικών byte και του κρατούμενου C=0 από την πρόσθεση των λιγότερο σημαντικών byte.**

| | |
|---|---|
| 00000001 | 256 |
| +00010000 | 4096 |
| +0 | +0 |
| $00010001_2$ | $4352_{10}$ |

**Κρατούμενο: C=0**

**Αποτέλεσμα: 000100001 00010010=4.352+18=4.370**

***Περίπτωση 2η. Πρόσθεση δύο αριθμών των 16 ψηφίων (δύο byte ο καθένας), με κρατούμενο απ' την πρόσθεση των λιγότερο σημαντικών byte.***

**$00000001\ 10000000_2 = 384_{10}$**

**$00000000\ 10000001_2 = 129_{10}$**

**Πρόσθεση των λιγότερο σημαντικών bytes (C = 0).**

**10000000=+128**

**+10000001=+129**

**Κρατούμενο: C=1** **$00000001_2 = 257_{10}$ (=256+1)**

***Περίπτωση 2η. Πρόσθεση δύο αριθμών των 16 ψηφίων (δύο byte ο καθένας), με κρατούμενο απ' την πρόσθεση των λιγότερο σημαντικών byte.***

**$00000001\ 10000000_2=384_{10}$**

**$00000000\ 10000001_2=129_{10}$**

**Πρόσθεση των περισσότερο σημαντικών byte και του κρατούμενου C=1 από την πρόσθεση των λιγότερο σημαντικών byte.**

**00000001=256**

**+00000000= 0**

**+1(=256)**

**Κρατούμενο: C=0** **$00000010_2=512_{10}$**

**Αποτέλεσμα=00000010 00000001=512+1=513**

## *Πρόσθεση δύο θετικών αριθμών χωρίς υπερχείλιση.*

**00000101=(+5)**

**+00000111=(+7)**

**Κρατούμενο: C=0** **$00001100_2=12_{10}$**

**Υπερχείλιση: V=0**

**Η παρουσία του "0" στο ψηφίο $d_7$ είναι ένδειξη ότι το αποτέλεσμα θα είναι θετικό. Παρατηρείται επίσης πως και οι δύο δείκτες παρέμειναν καθαροί.**

## *Πρόσθεση δύο θετικών αριθμών με υπερχείλιση.*

**01111111=(+127)**

**+00000010=(+2)**

**Κρατούμενο: C=0** **$10000001_2$=-127**

**Υπερχείλιση: V=1**

**Η παρουσία του "1" στο ψηφίο $d_7$ είναι ένδειξη ότι το αποτέλεσμα θα είναι αρνητικός αριθμός με μέγεθος ίσο με το συμπλήρωμα ¨ως προς 2¨ των υπόλοιπων εφτά ψηφίων δηλαδή, ο αριθμός $-127_{10}$. Ο δείκτης υπερχείλισης έγινε "1", δείχνοντας την υπέρβαση του επιτρεπόμενου ορίου για το αποτέλεσμα.**

***Πρόσθεση δύο θετικών αριθμών με υπερχείλιση.***

**01111111=(+127)**

**+00000010=(+2)**

**Κρατούμενο: C=0** **$10000001_2$=-127**

**Υπερχείλιση: V=1**

**Ο έλεγχος επομένως του δείκτη υπερχείλισης οδηγεί στο συμπέρασμα ότι το αποτέλεσμα στην πραγματικότητα δεν είναι αρνητικό, απλώς υπάρχει υπερχείλιση πάνω απ' την επιτρεπόμενη τιμή $+127_{10}$ ($129_{10}$). Επομένως ο χρήστης ειδοποιείται ότι το αποτέλεσμα είναι λάθος και θα πρέπει να χρησιμοποιήσει υπορουτίνα διόρθωσης.**

## *Πρόσθεση θετικού και αρνητικού αριθμού με θετικό αποτέλεσμα.*

**00000101=(+5)**

**+11111101 = (-3)**

**Κρατούμενο: C=1** **$00000010_2$=+2**

**Υπερχείλιση: V=0**

**Η παρουσία του "0" στο ψηφίο $d_7$ σημαίνει ότι το αποτέλεσμα είναι θετικό.**

## *Πρόσθεση θετικού και αρνητικού αριθμού με αρνητικό αποτέλεσμα.*

**00000101=(+5)**

**+11111001=(-7)**

**Κρατούμενο: C=0** $\quad$ $\mathbf{1}1111110_2 = -2$

**Υπερχείλιση: V=0**

**Η παρουσία του "1" στο ψηφίο $d_7$ σημαίνει ότι το αποτέλεσμα είναι αρνητικό.**

## *Πρόσθεση δύο αρνητικών αριθμών χωρίς υπερχείλιση.*

**11111011=(-5)**

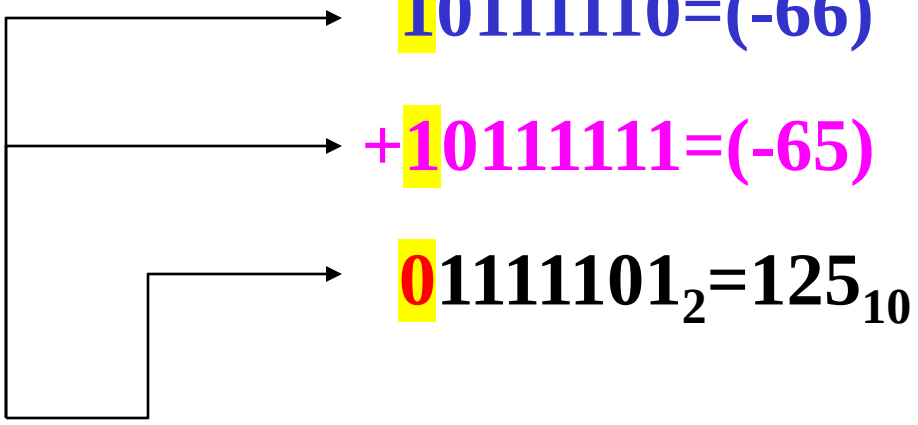
**+11111001=(-7)**

**Κρατούμενο: C=1** **11110100=-12**

**Υπερχείλιση: V=0**

**Η παρουσία του "1" στο ψηφίο $d_7$ σημαίνει ότι το αποτέλεσμα είναι αρνητικό.**

# *Πρόσθεση δύο αρνητικών αριθμών με υπερχείλιση.*




**Η παρουσία του "0" στο ψηφίο $d_7$ σημαίνει ότι το αποτέλεσμα θα είναι θετικός αριθμός, όμως ο δείκτης υπερχείλισης είναι "1" που σημαίνει πως ξεπεράστηκε το επιτρεπόμενο όριο. Χωρίς την ένδειξη της υπερχείλισης το αποτέλεσμα θα παρουσιαζόταν σαν +125. Η υπερχείλιση όμως δείχνει ότι τελικά το αποτέλεσμα είναι αρνητικό και μάλιστα έχει ξεπεράσει την τιμή -128.**

# Αφαίρεση Απλής και Πολλαπλής Ακρίβειας

**Η αφαίρεση στους υπολογιστές γίνεται μέσω πρόσθεσης με τη χρήση του «συμπληρώματος ως προς 2».**

**Κατά τα άλλα ισχύουν αυτά που αναφέρθηκαν στην πρόσθεση απλής και πολλαπλής ακρίβειας.**

# ADD.L D0,D7

**Πριν την εκτέλεση της εντολής**

**[D0]=$12345678, [D7]=$87654321**

**Μετά την εκτέλεση της εντολής**

**[D7]=$99999999**

**[D7]=[D0]+[D7]=$12345678+$87654321=$99999999**

# ADDI.L #$00001F00,D5

ADD IMMIDIATE

**Πριν την εκτέλεση της εντολής**

**[D5]=$12340078**

**Μετά την εκτέλεση της εντολής**

**[D5]=$12341F78**

**[D5]=**
**[D5]+$00001F00=$12340078+$00001F00=$12341F78**

# Κωδικός εντολής

**1. *ADDI (Πρόσθεσε τα απευθείας δεδομένα)***

**Δίνει το μνημονικό της εντολής ADDI με μια σύντομη περιγραφή του τι αυτό σημαίνει.**

**2. *Λειτουργία:* *Άμεσα δεδομένα + [Διεύθυνση προορισμού]***
***⇒ [Διεύθυνση προορισμού]***

**Δείχνει τη λειτουργία που καλείται να κάνει η εντολή.**

# Κωδικός εντολής

**3. *Σύνταξη συμβολομεταφραστή:***

***ADDI#<Δεδομένα>, <ΕΑ>***

**Δίνεται η απαιτούμενη από το συμβολομεταφραστή σύνταξη της εντολής.**

**Το # δηλώνει ότι τα <δεδομένα> που ακολουθούν είναι τα ίδια τα δεδομένα (τελεστέος προέλευσης) πάνω στα οποία θα γίνει η εκτέλεση της εντολής.**

**<ΕΑ> (Effective Address) είναι η Ενεργός Διεύθυνση (ΕΔ), που είναι ένας γενικός όρος για τη θέση μνήμης η οποία καθορίζεται από τη μέθοδο διευθυνσιοδότησης του τελεστέου προορισμού.**

**4. *Ιδιότητες:* *Μέγεθος=(Byte, Λέξη, Μακριά λέξη)***

**Αναφέρονται τα μήκη των ψηφιολέξεων που μπορεί να επεξεργαστεί η συγκεκριμένη εντολή.**

**5. *Περιγραφή:* *Προσθέτει τα απευθείας δεδομένα στον τελεστέο προορισμού, και αποθηκεύει το αποτέλεσμα στη θέση προορισμού. Το μέγεθος της λειτουργίας μπορεί να καθοριστεί ως byte, λέξη ή μακριά λέξη. Το μέγεθος των απευθείας δεδομένων ταιριάζει με το μέγεθος της λειτουργίας.***

**Δίνεται μια πιο λεπτομερής περιγραφή του τι κάνει ο μικροεπεξεργαστής όταν εκτελεί τη συγκεκριμένη εντολή.**

## *6. Κώδικες συνθήκης ή Δείκτες ή Σημαίες του καταχωρητή κατάστασης:*

**Δίνεται ο τρόπος με τον οποίο επηρεάζονται οι κώδικες ή δείκτες ή σημαίες του καταχωρητή κατάστασης όταν εκτελείται η συγκεκριμένη εντολή.**

| X | N | Z | V | C |
|---|---|---|---|---|
| • | • | • | • | • |

***X Παίρνει την ίδια τιμή με το ψηφίο κρατούμενου.***

***N Παίρνει την τιμή ¨1¨ όταν το αποτέλεσμα είναι αρνητικός αριθμός. Διαφορετικά παίρνει την τιμή ¨0¨.***

## *6. Κώδικες συνθήκης ή Δείκτες ή Σημαίες του καταχωρητή κατάστασης:*

| X | N | Z | V | C |
|---|---|---|---|---|
| ● | ● | ● | ● | ● |

***Z** Παίρνει την τιμή ¨1¨ όταν το αποτέλεσμα είναι μηδέν. Διαφορετικά παίρνει την τιμή ¨0¨.*

***V** Παίρνει την τιμή ¨1¨ όταν γεννάται υπερχείλιση. Διαφορετικά παίρνει την τιμή ¨0¨.*

***C** Παίρνει την τιμή ¨1¨ όταν γεννάται κρατούμενο. Διαφορετικά παίρνει την τιμή ¨0¨.*

# 7. Διάταξη Εντολής:

| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ΜΕΓΕΘΟΣ | | ΕΝΕΡΓΟΣ ΔΙΕΥΘΥΝΣΗ | | | | | |
| | | | | | | | | | | ΜΟΡΦΗ | | | ΚΑΤΑΧΩΡΗΤΗΣ | | |
| ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (16 ΨΗΦΙΑ) | | | | | | | | ΒΥΤΕ ΔΕΔΟΜΕΝΩΝ (8 ΨΗΦΙΑ) | | | | | | | |
| ΜΑΚΡΥΑ ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (32 ΨΗΦΙΑ) | | | | | | | | | | | | | | | |

**Δίνεται η διάταξη της εντολής, που δείχνει ποιες τιμές ψηφίων χρησιμοποιούνται για να αναπαραστήσουν την εντολή στο δυαδικό σύστημα.**

**Τα ψηφία $d_{15}$-$d_8$ έχουν σταθερές τιμές για όλες τις εκδοχές της εντολής ADDI ενώ τα ψηφία και $d_7$-$d_0$ συμπληρώνονται κατά περίπτωση σύμφωνα με το σημείο 8 που αναφέρεται παρακάτω.**

# 8. *Πεδία εντολής:*

| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ΜΕΓΕΘΟΣ | | ΕΝΕΡΓΟΣ ΔΙΕΥΘΥΝΣΗ | | | | | |
| | | | | | | | | | | ΜΟΡΦΗ | | | ΚΑΤΑΧΩΡΗΤΗΣ | | |
| ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (16 ΨΗΦΙΑ) | | | | | | | | ΒΥΤΕ ΔΕΔΟΜΕΝΩΝ (8 ΨΗΦΙΑ) | | | | | | | |
| ΜΑΚΡΥΑ ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (32 ΨΗΦΙΑ) | | | | | | | | | | | | | | | |

***Πεδίο μεγέθους-Καθορίζει το μέγεθος της λειτουργίας:***

***00-Λειτουργία Byte***

***01-Λειτουργία Λέξης***

***10-Λειτουργία Μακριάς λέξης***

**Δίνονται τα ψηφία που καθορίζουν το μέγεθος των δεδομένων που καλείται να επεξεργαστεί η εντολή.**

| $d_{15}$ | $d_{14}$ | $d_{13}$ | $d_{12}$ | $d_{11}$ | $d_{10}$ | $d_9$ | $d_8$ | $d_7$ | $d_6$ | $d_5$ | $d_4$ | $d_3$ | $d_2$ | $d_1$ | $d_0$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | ΜΕΓΕΘΟΣ | | ΕΝΕΡΓΟΣ ΔΙΕΥΘΥΝΣΗ | | | | | |
| | | | | | | | | | | ΜΟΡΦΗ | | | ΚΑΤΑΧΩΡΗΤΗΣ | | |
| ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (16 ΨΗΦΙΑ) | | | | | | | | ΒΥΤΕ ΔΕΔΟΜΕΝΩΝ (8 ΨΗΦΙΑ) | | | | | | | |
| ΜΑΚΡΥΑ ΛΕΞΗ ΔΕΔΟΜΕΝΩΝ (32 ΨΗΦΙΑ) | | | | | | | | | | | | | | | |

***Πεδίο Ενεργού Διεύθυνσης-Καθορίζει τον τελεστέο προορισμού***

***Επιτρέπονται μόνο εναλλακτικές μέθοδοι διευθυνσιοδότησης.***

***Απευθείας πεδίο-(Τα δεδομένα ακολουθούν αμέσως μετά την εντολή).***

***Αν το μέγεθος=00, τα δεδομένα είναι το χαμηλότερο byte της απευθείας λέξης.***

***Αν το μέγεθος=01, τα δεδομένα είναι το σύνολο της απευθείας λέξης.***

***Αν το μέγεθος=10, τα δεδομένα είναι οι επόμενες δύο απευθείας λέξεις.***

# ADDQ #7,D0 ADD QUICK

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000100**

**Μετά την εκτέλεση της εντολής**

**[D0]=$00000107**

**[D0]=[D0]+7=$00000100+7=$00000107**

ADD QUICK

Syntax: ADDQ #<data>, <ea>, Size = byte, word, longword

* The immediate data must be in the range 1 to 8

** Note that a word operation on an address register affects all bits of the register

# ADDX.L D0,D5

ADD EXTENDED

**Πριν την εκτέλεση της εντολής**

**[D0]=$01234567, [D5]=$76543210, X=1**

**Μετά την εκτέλεση της εντολής**

**[D5]=$77777778**

**[D5]=[D5]+[D0]+X=$76543210+$01234567+1=$77777778**

# ADDX.L -(A0),-(A5)

**Πριν την εκτέλεση της εντολής**

**[A0]=$00401004, [A5]=$00400504,**

**[$401000]=$00401501, [$400500]=$00001100, X=1**

**Μετά την εκτέλεση της εντολής**

**[(A5)]=$00402602**

**[(A5)]=[$00401000]+[$00400500]+X**
**=$00401501+$00001100+1=$00402602**

# ADDA.L D0,A5 ADD ADDRESS

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000100 [A5]=$00400500**

**Μετά την εκτέλεση της εντολής**

**[A5]=$00400600**

**[A5]=$00400500+$00000100=$00400600**

ADD ADDRESS

Syntax: ADDA <ea>,An Size = word, longword

* If it is a word, it is sign-extended to a longword

** The conditions codes are not affected

# SUB.L D0,D7

**Πριν την εκτέλεση της εντολής**

**[D0]=$12345678, [D7]=$99999999**

**Μετά την εκτέλεση της εντολής**

**[D7]=$87654321**

**[D7]=[D7]-[D0]=$99999999-$12345678=$87654321**

# SUBI.W #$01FF,D5

**Πριν την εκτέλεση της εντολής**

**[D5]=$1234FFFF**

**Μετά την εκτέλεση της εντολής**

**[D5]=$1234FE00**

**[D5]=[D5]-$01FF=$1234FFFF-$01FF=$1234FE00**

# SUBX.L D0,D5

**Πριν την εκτέλεση της εντολής**

**[D0]=$01234567, [D5]=$77777778, X=1**

**Μετά την εκτέλεση της εντολής**

**[D5]=$76543210**

**[D5]=[D5]-[D0]-X=$77777778-$01234567-1=$76543210**

# SUBQ #2,D0

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000102**

**Μετά την εκτέλεση της εντολής**

**[D0]=$00000100**

**[D0]=[D0]-2=$00000102-2=$00000100**

# SUBX.L -(A0),-(A5)

**Πριν την εκτέλεση της εντολής**

**[A0]=$00401004, [A5]=$00400504,**

**[$401000]=$00001100, [$400500]=$00402501, X=1**

**Μετά την εκτέλεση της εντολής**

**[(A5)]=$00401400**

**[(A5)]=[$00405000]-[$00001100]-X**
**=$00402501-$00001100-1=$00401400**

# SUBA.L D0,A5

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000100 [A5]=$00400600**

**Μετά την εκτέλεση της εντολής**

**[A5]=$00400500**

**[A5]=$00400600-$00000100=$00400500**

# NEG.W D0

NEGATE

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000FFF [$4.095_{10}$]**

**Μετά την εκτέλεση της εντολής**

**[D0]=$0000F001**

**[D0]=$00000000-[D0]=$00000000-$00000FFF=$0000F001**

**[$-4.095_{10}$]**

# NEGX.W D0

NEGATE WITH EXTEND

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000FFF, X=1**

**[$4.095_{10}$]**

**Μετά την εκτέλεση της εντολής**

**[D0]=$0000F000**

**[D0]=$00000000-[D0]-X=$00000000-$00000FFF-1**
**=$0000F000**

**[$-4.096_{10}$]**

# Εντολές Σύγκρισης και Ελέγχου

# CMP.L D0,D4

μόνο Dn

**Πριν την εκτέλεση της εντολής**

**[D0]=$22345678, [D4]=$99999999**

**Μετά την εκτέλεση της εντολής**

**[D0]=$22345678,**
**[D4]=$99999999**

**[D4]-[D0]= $99999999 - $22345678=$77654321**

**= 0111 0111 0110 0101 0100 0011 0010 0001**

**= Θετικός αριθμός, διάφορος του μηδενός**

| T | | S | | | I2 | I1 | I0 | | | | X | N | Z | V | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | 0 | 0 | 0 | 1 | 0 |

# Πιο αναλυτικά η CMP…

**[D0]=$22345678 = 0010 0010 0011 0100 0101 0110 0111 1000**
**[D4]=$99999999 = 1001 1001 1001 1001 1001 1001 1001 1001**

**Η εντολή `CMP.L D0,D4`**

ο M68000 θα εκτελέσει πρόσθεση συμπληρώματος ως προς 2 μεταξύ D4 και D0.

`D4-D0 = D4 + [NOT(D0)+1]`

**[D4]= $99999999 = 1001 1001 1001 1001 1001 1001 1001 1001**

**[NOT(D0)+1]]=$DDCBA988 = 1101 1101 1100 1011 1010 1001 1000 1000**

---

**RESULT=$77654321 = 0111 0111 0110 0101 0100 0011 0010 0001**

ο SR έχει τα εξής CCR bits:

| | T | S | | INT | | | XNZVC |
|---|---|---|---|---|---|---|---|
| SR= | 0 | 0 | 1 | 0 | 0 | 0 | 0000000000010 |

N=0 το αποτέλεσμα είναι θετικό.
Z=0 το αποτέλεσμα είναι μη μηδενικό.
V=1 γιατί έχουμε D4 και D0 με το ίδιο πρόσημο (1) ενώ το αποτέλεσμα έχει διαφορετικό πρόσημο (0). Άρα έχουμε υπερχείλιση.
C=0 γιατί ο D4 > D0, θα είχαμε C=1 αν D4 < D0.

# CMPA.L (A0),A4

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400500, [$400500]=$004033FF, [A4]=$004033FF**

**Μετά την εκτέλεση της εντολής**

**[$400500]=$004033FF, [A4]=$004033FF**

**[A4]-[$00400500]=$004033FF-$004033FF =$00000000**

| T | | S | | | I2 | I1 | I0 | | | | X | N | Z | V | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | 0 | 0 | 1 | 0 | 0 |

If it is a word, it is sign-extended to 32-bit before the comparison

# CMPI.B #$F3,D0

**Πριν την εκτέλεση της εντολής**

**[D0]=$172B23F3**

**Μετά την εκτέλεση της εντολής**

**[D0]=$172B23F3**

**[LSB D0]-[$F3]=$F3-$F3 =$00**

| T | | S | | | I2 | I1 | I0 | | | | X | N | Z | V | C |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | | 0 | 0 | 1 | 0 | 0 |

# Εντολές Διακλάδωσης

| Instruction | Description |
|---|---|
| **BRA** | **BRA (branch always)** implements an unconditional branch, relative to the PC. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, BRA cannot be used. |
| **JMP** | **JMP (jump)** is like BRA. The only difference is that BRA uses only relative addressing, whereas JMP has more addressing modes, including absolute address. |
| **Bcc** | **Bcc (branch on condition code)** is used whenever program execution must follow one of two paths depending on a condition. The condition is specified by the mnemonic **cc**. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, Bcc cannot be used. |
| **JSR**<br>**BSR**<br>**RTS**<br>**RTE**<br>**RTI** | **JSR and BSR** branches to a subroutine. The PC is saved on the stack before loading the PC with the new value.<br>RTS is used to return from the subroutine by restoring the PC from the stack. |

| | cc | Condition | Branch Taken If |
|---|---|---|---|
| | CC | Carry clear | C = 0 |
| | CS | Carry set | C = 1 |
| | NE | Not equal | Z = 0 |
| | EQ | Equal | Z = 1 |
| | PL | Plus | N = 0 |
| | MI | Minus | N = 1 |
| | VC | Overflow clear | V = 0 |
| | VS | Overflow set | V = 1 |
| SIGNED | GE | Greater or equal | N'V + NV' = 0 |
| SIGNED | GT | Greater than | NVZ + (NVZ)' = 1 |
| SIGNED | LE | Less or equal | Z+(N'V+NV') = 1 |
| SIGNED | LT | Less than | N'V + NV' = 1 |
| UNSIGNED | HS | Higher or same | C = 0 |
| UNSIGNED | LO | Lower | C = 1 |
| UNSIGNED | HI | Higher | C'Z' = 1 |
| UNSIGNED | LS | Lower or same | C + Z=1 |

# BRA.S LABEL (για byte)
# BRA LABEL (για word)

**Πριν την εκτέλεση της εντολής**

**[Παρών ΜΠ]=$XXXXXXXX**

**Μετά την εκτέλεση της εντολής**

**[Νέος ΜΠ]=[Παρών ΜΠ]+offset**

**Το offset είναι ένας προσημασμένος αριθμός μήκους byte ή μήκους word (8 ή 16 ψηφίων).**

**Αν το offset είναι μήκους byte μπορεί να προκαλέσει μετατόπιση από (-128)-(+127) byte.**

**Αν το offset είναι μήκους word μπορεί να προκαλέσει μετατόπιση από (-32.768)-(+32.767) byte.**

# JMP (A1)

**Πριν την εκτέλεση της εντολής**

**[A1]=$0040D641, [ΜΠ]=$XXXXXXXX**

**Μετά την εκτέλεση της εντολής**

**[ΜΠ]=$0040D641**

# JMP (A0,D0.L)

**Μετά την εκτέλεση της εντολής**

JUMP (UNCONDITIONALLY) **[ΜΠ]= [A0] + [D0] (offset)**

Syntax: JMP <ea> Unsized

* Useful command for calculation (at compile time)
of dynamic or computed jumps to an address (ea)

** The conditions codes are not affected

# BCS.S LABEL (για byte)
# BCS LABEL (για word)

**Πριν την εκτέλεση της εντολής**

**[Παρών ΜΠ]=$XXXXXXXX**

**Μετά την εκτέλεση της εντολής**

**Αν C=1**

**[Νέος ΜΠ]=[Παρών ΜΠ]+offset**

**Αν C=0**

**[Νέος ΜΠ]=[Παρών ΜΠ]**

# Εντολές Ελέγχου συνθήκης Μείωσης και Διακλάδωσης

# DBcc Dn,<label>

**Η εντολή θα μειώνει το περιεχόμενο του Dn έως ότου γίνει -1 που τερματίζει και τη διακλάδωση στο <label>.**

**Η εντολή έχει την αντίθετη λειτουργία με τις Bcc εντολές. Δηλαδή για παράδειγμα η BCC κάνει διακλάδωση αν το κρατούμενο είναι μηδέν, ενώ η DBCC τερματίζει τον βρόγχο όταν το κρατούμενο είναι μηδέν.**

**Τέλος, οι DBcc είναι κατάλληλες για διαχείριση επαναλαμβανόμενων δομών (REPEAT…UNTIL). Επειδή, κάνει έλεγχο της συνθήκης (τιμή του Dn, αν έχει την τιμή -1), μειώνει τον μετρητή (Dn) και διακλαδώνει ή τερματίζει την επαναληπτική δομή.**

# DBRA Dn,LABEL

**Η εντολή θα μειώνει το περιεχόμενο του Dn και:**

**Α. Για όσο δεν είναι –1 θα διακλαδώνει στην ετικέτα LABEL.**

**Β. Όταν το περιεχόμενο του Dn γίνει –1 θα συνεχίσει με την εκτέλεση της επόμενης στη λίστα εντολής.**

UNCONDITIONAL RELATIVE BRANCH (goto) WITH DECREMENT
Syntax: DBRA Dn, <label>  Size = byte, word
* Program execution continues at location [PC]+d,
** d: two's complement value  *** PC value = PCcurrent + 2

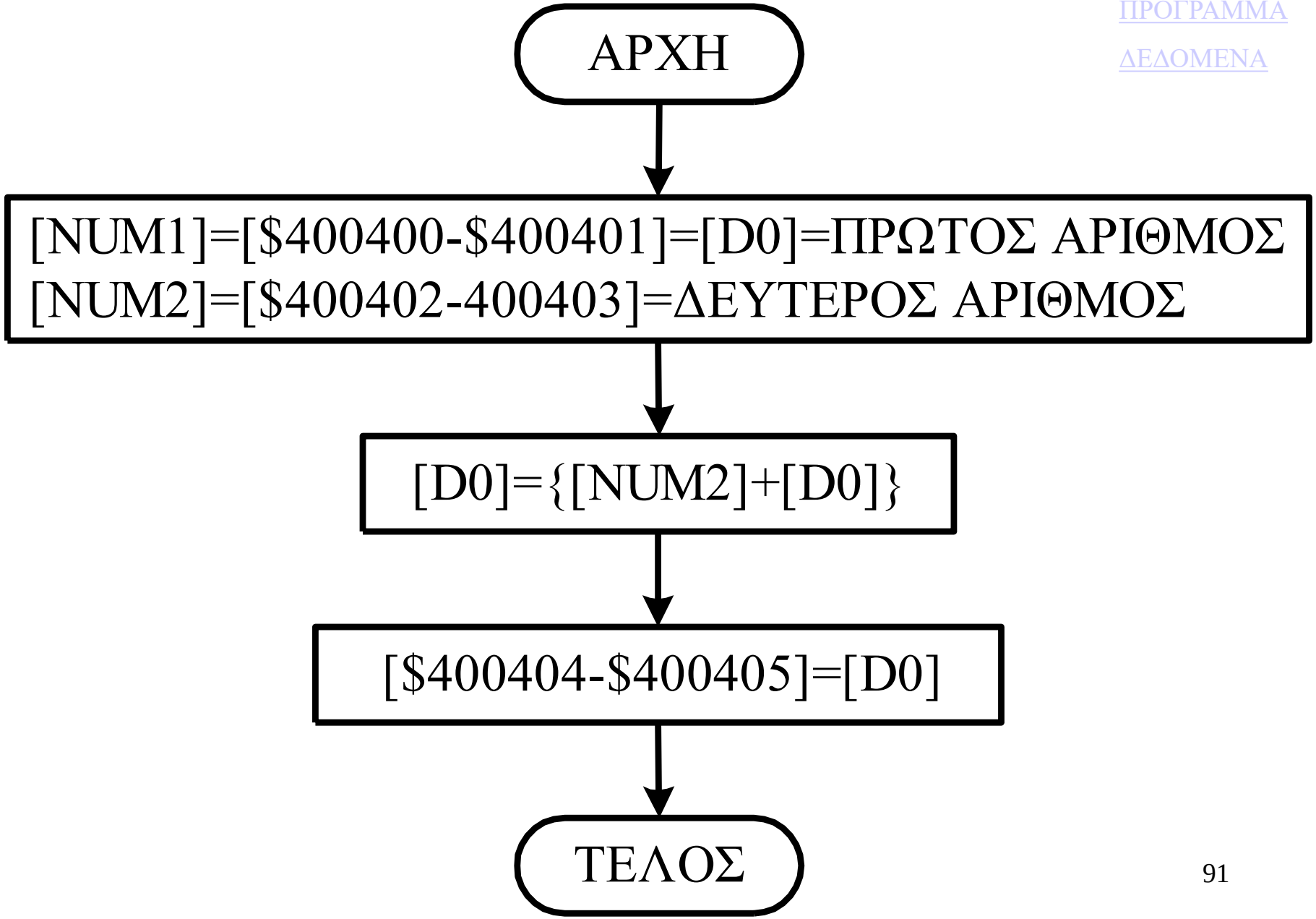
# Παράδειγμα 3.8



***Να γραφτεί μια υπορουτίνα που θα προσθέτει δύο αριθμούς μήκους λέξης που βρίσκονται στις θέσεις μνήμης $400400 και $400402 αντίστοιχα και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400404 (θεωρείστε ότι το αποτέλεσμα είναι αριθμός μικρότερος του 65535).***

# Παράδειγμα 3.8

ΕΚΦΩΝΗΣΗ
ASSEMBLY
ΠΡΟΓΡΑΜΜΑ
ΔΕΔΟΜΕΝΑ



```
        ORG $400400
NUM1    DC.W $1234
NUM2    DC.W $5432
SUM     DS.W  1

        ORG $400410
SUBRTN  MOVE.W NUM1,D0
        ADD.W NUM2,D0
        MOVE.W D0,SUM
        END SUBRTN
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
 1  00400400                          ORG       $400400
 2  00400400 1234          NUM1:      DC.W      $1234
 3  00400402 5432          NUM2:      DC.W      $5432
 4  00400404 00000002      SUM:       DS.W      1
 5
 6  00400410                          ORG       $400410
 7  00400410 303900400400  SUBRTN:    MOVE.W    NUM1,D0
 8  00400416 D07900400402              ADD.W     NUM2,D0
 9  0040041C 33C000400404              MOVE.W    D0,SUM
10  00400410                          END       $400410
```

ΕΚΦΩΝΗΣΗ

ΛΔ

ΠΡΟΓΡΑΜΜΑ

ΔΕΔΟΜΕΝΑ

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 12 34 54 32 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 30 39 00 40 04 00 D0 79 00 40 04 02 33 C0 00 40**

**400420 04 04 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος

**400400 12 34 54 32 66 66 00 00 00 00 00 00 00 00 00 00**

ΕΚΦΩΝΗΣΗ

ΛΔ

ΠΡΟΓΡΑΜΜΑ

ASSEMBLY

# Παράδειγμα

***Να γραφτεί ένα πρόγραμμα που θα προσθέτει τους αριθμούς 1, 2 και 3 που είναι αποθηκευμένοι στη μνήμη που αρχίζει από τη θέση NUMS και θα αποθηκεύει το αποτέλεσμα σε μορφή μακριάς λέξης στη θέση μνήμης RESULT.***

# Παράδειγμα

```
          ORG $400440
COUNT     DC.B $3
NUMS      DC.B $1,$2,$3
RESULT    DS.L 1

          ORG $400400
START     CLR D0
          LEA NUMS,A0
          MOVE.B COUNT,D1
LOOP      ADD.B (A0)+,D0
          SUBQ #1,D1
          BNE LOOP
          MOVE.L D0,RESULT
          END START
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                          ORG       $400400
 2  00400400 03          COUNT:       DC.B      3
 3  00400401 010203      NUMS:        DC.B      1,2,3
 4  00400404 00000004    RESULT:      DS.L      1
 5  00400410                          ORG       $400410
 6  00400410 4280        SUBRTN:      CLR.L     D0
 7  00400412 41F900400401             LEA       NUMS,A0
 8  00400418 123900400400             MOVE.B    COUNT,D1
 9  0040041E D018        LOOP:        ADD.B     (A0)+,D0
10  00400420 5341                     SUBQ      #1,D1
11  00400422 66FA                     BNE       LOOP
12  00400424 23C000400404             MOVE.L    D0,RESULT
13  00400410                          END       $400410
```

**Τα περιεχόμενα μνήμης πριν απ' την εκτέλεση του προγράμματος θα είναι:**

**400400 03 01 02 03 00 00 00 00 00 00 00 00 00 00 00 00**

**400410 42 80 41 F9 00 40 04 01 12 39 00 40 04 00 D0 18**

**400420 53 41 66 FA 23 C0 00 40 04 04 00 00 00 00 00 00**

**Τα περιεχόμενα μνήμης μετά την εκτέλεση του προγράμματος θα είναι:**

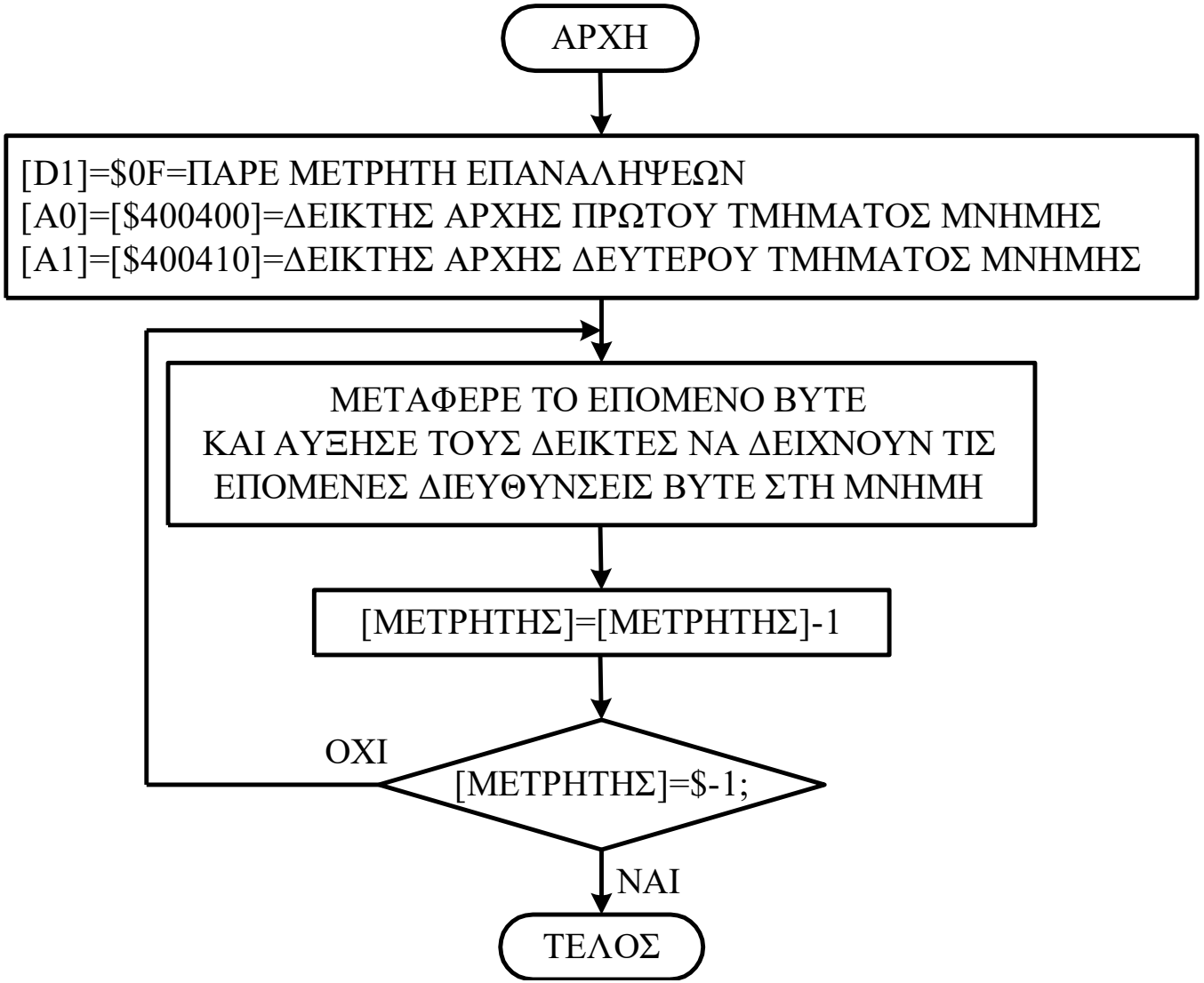
**400400 03 01 02 03 00 00 00 06 00 00 00 00 00 00 00 00**

**400410 42 80 41 F9 00 40 04 01 12 39 00 40 04 00 D0 18**

**400420 53 41 66 FA 23 C0 00 40 04 04 00 00 00 00 00 00**

# Παράδειγμα 3.7

***Να γραφτεί μια υπορουτίνα που θα μεταφέρει τα περιεχόμενα των θέσεων μνήμης $400400-$40040F στις θέσεις μνήμης $400410-$40041F ένα byte κάθε φορά.***



```
        ORG $400400
TABLE1  DC.B 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
TABLE2  DS.B $10

        ORG $400420
SUBRTN  MOVE.B #$0F,D1
        LEA TABLE1,A0
        LEA TABLE2,A1
LOOP    MOVE.B (A0)+,(A1)+
        DBRA D1, LOOP
        END $400420
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                          ORG       $400400
 2  00400400
    00 01 02 03 04 05      TABLE1:    DC.B      0,1,2,3,4,5,6,7,8,9,
    06 07 08 09 0A 0B                            10,11,12,13,14,15
    0C 0D 0E 0F
 3  00400410 00000010      TABLE2:    DS.B      $10
 4
 5  00400420                          ORG       $400420
 6  00400420 123C000F      SUBRTN:    MOVE.B    #$0F,D1
 7  00400424 41F900400400             LEA       TABLE1,A0
 8  0040042A 43F900400410             LEA       TABLE2,A1
 9  00400430 12D8          LOOP:      MOVE.B    (A0)+,(A1)+
10  00400432 51C9FFFC                 DBRA      D1,LOOP
11  00400420                          END       $400420
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

**400410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400420 12 3C 00 0F 41 F9 00 40 04 00 43 F9 00 40 04 10**

**400430 12 D8 51 C9 FF FC 00 00 00 00 00 00 00 00 00 00**

α. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

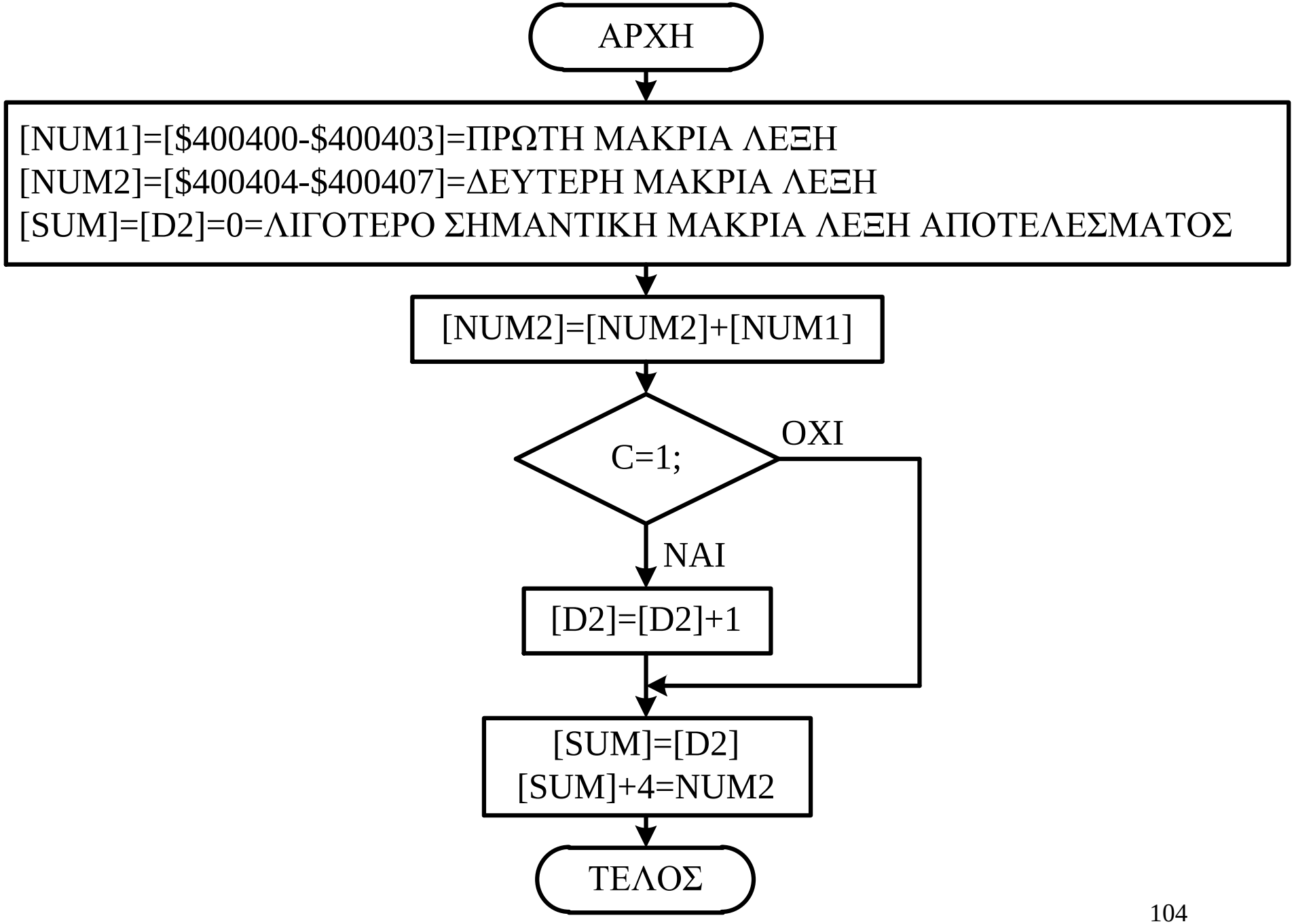
**400400 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

**400410 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F**

# Παράδειγμα 3.9

***Να γραφτεί μια υπορουτίνα που θα προσθέτει τη μακριά λέξη που είναι αποθηκευμένη στη θέση μνήμης $400400, στη μακριά λέξη που είναι αποθηκευμένη στη θέσης μνήμης $400404. Το αποτέλεσμα να αποθηκευτεί στις θέσεις μνήμης $400408 και $40040C.***

***(Η περισσότερο σημαντική μακριά λέξη του αθροίσματος στη θέση $400408).***


ΑΡΧΗ
[NUM1]=[$400400-$400403]=ΠΡΩΤΗ ΜΑΚΡΙΑ ΛΕΞΗ
[NUM2]=[$400404-$400407]=ΔΕΥΤΕΡΗ ΜΑΚΡΙΑ ΛΕΞΗ
[SUM]=[D2]=0=ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΑΠΟΤΕΛΕΣΜΑΤΟΣ
[NUM2]=[NUM2]+[NUM1]
C=1;
ΟΧΙ
ΝΑΙ
[D2]=[D2]+1
[SUM]=[D2]
[SUM]+4=NUM2
ΤΕΛΟΣ

```
                ORG $400400
NUM1            DC.L $11111112
NUM2            DC.L $EEEEEEEF
SUM             DS.L  2

                ORG $400410
SUBRTN          CLR  D2
                MOVE.L NUM1,D0
                MOVE.L NUM2,D1
                ADD.L D0,D1
                BCC  NOCARRY
                MOVEQ  #1,D2
NOCARRY         MOVE.L D2,SUM
                MOVE.L D1,SUM+4
                END SUBRTN
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
1  00400400                          ORG       $400400
2  00400400 11111112      NUM1:      DC.L      $11111112
3  00400404 EEEEEEEF      NUM2:      DC.L      $EEEEEEEF
4  00400408 00000008      SUM:       DS.L      2
5
6  00400410                          ORG       $400410
7  00400410 4282          SUBRTN:    CLR.L     D2
8  00400412 203900400400             MOVE.L    NUM1,D0
9  00400418 223900400404             MOVE.L    NUM2,D1
10 0040041E D280                     ADD.L     D0,D1
11 00400420 64000004                 BCC       NOCARRY
12 00400424 7401                     MOVEQ     #1,D2
13 00400426 23C200400408 NOCARRY:    MOVE.L    D2,SUM
14 0040042C 23C10040040C             MOVE.L    D1,SUM+4
15 00400410                          END       $400410
```

ΔΔ
ΠΡΟΓΡΑΜΜΑ
ΕΚΦΩΝΗΣΗ
ΔΕΔΟΜΕΝΑ

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 11 11 11 12 EE EE EE EF 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 42 82 20 39 00 40 04 00 22 39 00 40 04 04 D2 80**

**400420 64 00 00 04 74 01 23 C2 00 40 04 08 23 C1 00 40**

**400430 04 0C 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 11 11 11 12 EE EE EE EF 00 00 00 01 00 00 00 01**

ΛΔ

ΠΡΟΓΡΑΜΜΑ

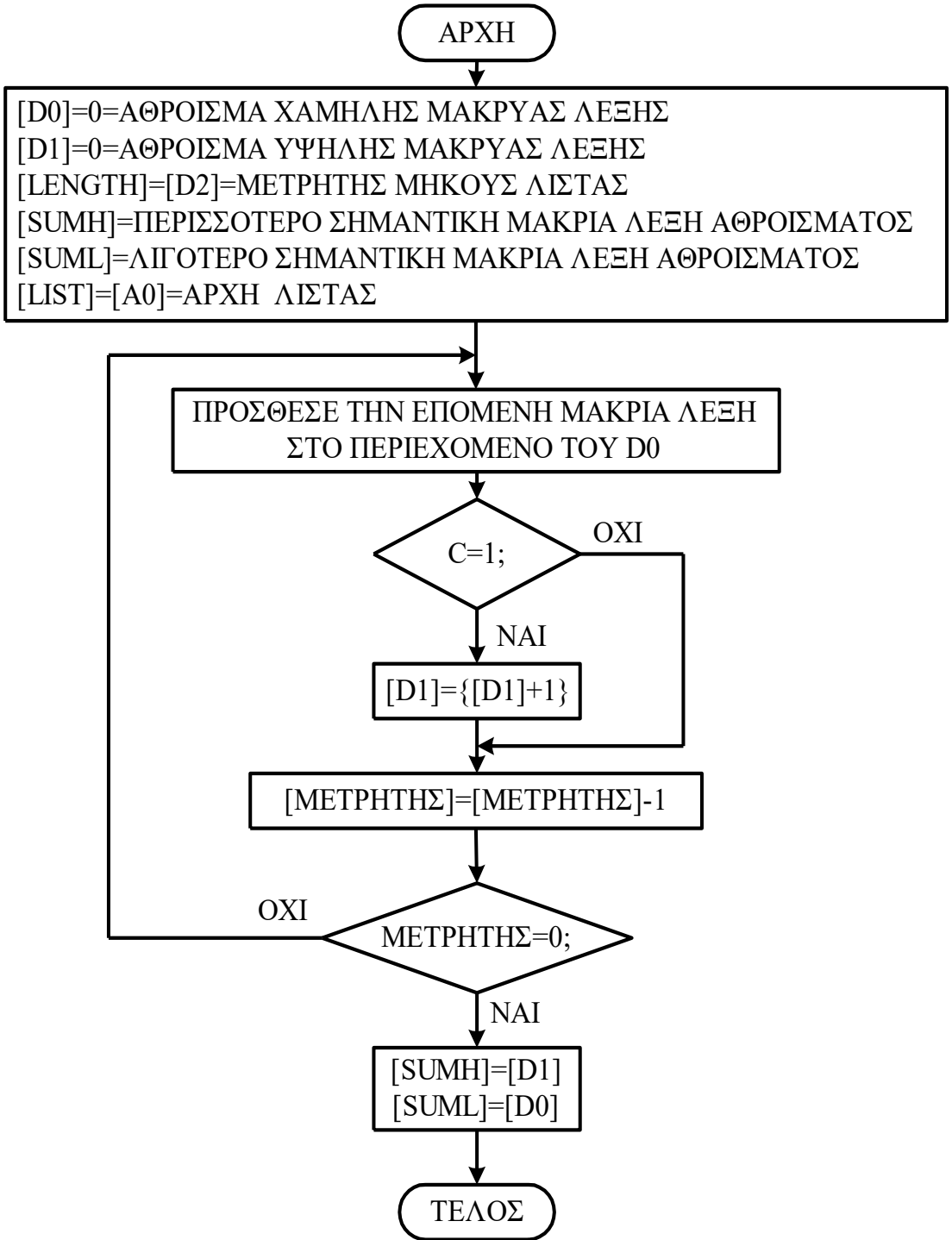
ΕΚΦΩΝΗΣΗ

ASSEMBLY

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που να υπολογίζει το άθροισμα μιας σειράς αριθμών μήκους μακριάς λέξης. Το μήκος της σειράς βρίσκεται στις θέσεις μνήμης $400400-$400403 και η σειρά αρχίζει από τη θέση μνήμης $40040C. Το άθροισμα να αποθηκεύεται στις θέσεις μνήμης $400404-$40040B.***

***(Η περισσότερο σημαντική μακριά λέξη του αθροίσματος στις θέσεις $400404).***


ΑΡΧΗ
[D0]=0=ΑΘΡΟΙΣΜΑ ΧΑΜΗΛΗΣ ΜΑΚΡΥΑΣ ΛΕΞΗΣ
[D1]=0=ΑΘΡΟΙΣΜΑ ΥΨΗΛΗΣ ΜΑΚΡΥΑΣ ΛΕΞΗΣ
[LENGTH]=[D2]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[SUMH]=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΑΘΡΟΙΣΜΑΤΟΣ
[SUML]=ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΑΘΡΟΙΣΜΑΤΟΣ
[LIST]=[A0]=ΑΡΧΗ ΛΙΣΤΑΣ
ΠΡΟΣΘΕΣΕ ΤΗΝ ΕΠΟΜΕΝΗ ΜΑΚΡΙΑ ΛΕΞΗ
ΣΤΟ ΠΕΡΙΕΧΟΜΕΝΟ ΤΟΥ D0
C=1;
ΟΧΙ
ΝΑΙ
[D1]={[D1]+1}
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
ΟΧΙ
ΜΕΤΡΗΤΗΣ=0;
ΝΑΙ
[SUMH]=[D1]
[SUML]=[D0]
ΤΕΛΟΣ

```
                ORG $400400
LENGTH          DC.L 6
SUMH            DS.L 1
SUML            DS.L 1
LIST            DC.L 1,1,1,1,$7FFFFFF,$80000000

                ORG $400430
SUBRTN          CLR.L D1
                CLR.L D0
                MOVE.L LENGTH,D2
                LEA LIST,A0
LOOP            ADD.L (A0)+,D0
                BCC NOCARRY
                ADDQ.L #1,D1
NOCARRY         SUBQ.L #1,D2
                BNE LOOP
                MOVE.L D1,SUMH
                MOVE.L D0,SUML
                END SUBRTN
```

```
1  00400400                                ORG       $400400
2  00400400 00000006          LENGTH:      DC.L      6
3  00400404 00000004          SUMH:        DS.L      1
4  00400408 00000004          SUML:        DS.L      1
5  0040040C 000000010000      LIST:        DC.L      1,1,1,1,
            00010000000 1                            00000001,
            7FFFFFF80000000                          $7FFFFFF,
                                                     $80000000
6
7  00400430                                ORG       $400430
8  00400430 4281              SUBRTN:      CLR.L     D1
9  00400432 4280                           CLR.L     D0
10 00400434 243900400400                   MOVE.L    LENGTH,D2
11 0040043A 41F900400404                   LEA       LIST,A0
12 00400440 D098              LOOP:        ADD.L     (A0)+,D0
13 00400442 64000004                       BCC        NOCARRY
14 00400446 5241                           ADDQ      #1,D1
15 00400448 5342              NOCARRY:     SUBQ      #1,D2
16 0040044A 66F4                           BNE       LOOP
17 0040044C 23C10040041C      ΛΔ           MOVE.L    D1,SUMH
18 00400452 23C000400420      ΠΡΟΓΡΑΜΜΑ    MOVE.L    D0,SUML
19 00400430                   ΕΚΦΩΝΗΣΗ     END       $400430
                              ΔΕΔΟΜΕΝΑ
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 00 00 00 06 00 00 00 00 00 00 00 00 00 00 00 01**
**400410 00 00 00 01 00 00 00 01 00 00 00 01 7F FF FF FF**
**400420 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400430 42 81 42 80 24 39 00 40 04 00 41 F9 00 40 04 04**
**400440 D0 98 64 00 00 04 52 41 53 42 66 F4 23 C1 00 40**
**400450 04 1C 23 C0 00 40 04 20 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

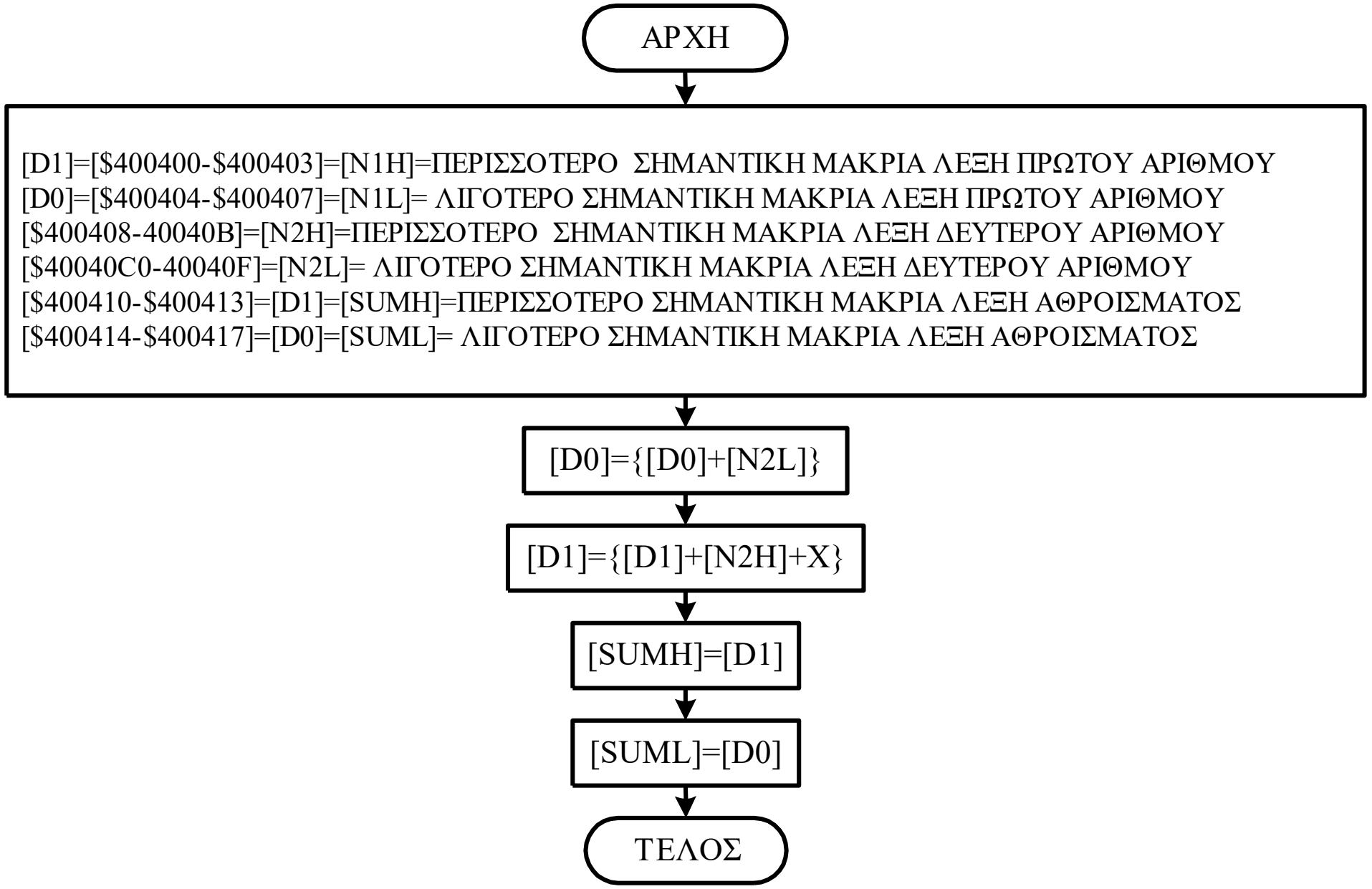
**400400 00 00 00 06 00 00 00 00 88 00 00 03 00 00 00 01**
**400410 00 00 00 01 00 00 00 01 00 00 00 01 7F FF FF FF**
**400420 80 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που θα κάνει πρόσθεση πολλαπλής ακρίβειας προσθέτοντας στον αριθμό 64 ψηφίων (δύο μακριές λέξεις) που είναι αποθηκευμένος στις θέσεις μνήμης $400400-$400407, τον αριθμό 64 ψηφίων (δύο μακριές λέξεις) που είναι αποθηκευμένος στις θέσεις μνήμης $400408-$40040F.***

***Το αποτέλεσμα να αποθηκευτεί στις θέσεις μνήμης $400410-$400417.***

***(Η περισσότερο σημαντική μακριά λέξη βρίσκεται στις υψηλότερες θέσεις μνήμης).***


ΑΡΧΗ
[D1]=[$400400-$400403]=[N1H]=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΠΡΩΤΟΥ ΑΡΙΘΜΟΥ
[D0]=[$400404-$400407]=[N1L]= ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΠΡΩΤΟΥ ΑΡΙΘΜΟΥ
[$400408-40040B]=[N2H]=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΔΕΥΤΕΡΟΥ ΑΡΙΘΜΟΥ
[$40040C0-40040F]=[N2L]= ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΔΕΥΤΕΡΟΥ ΑΡΙΘΜΟΥ
[$400410-$400413]=[D1]=[SUMH]=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΑΘΡΟΙΣΜΑΤΟΣ
[$400414-$400417]=[D0]=[SUML]= ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΗ ΜΑΚΡΙΑ ΛΕΞΗ ΑΘΡΟΙΣΜΑΤΟΣ
[D0]={[D0]+[N2L]}
[D1]={[D1]+[N2H]+X}
[SUMH]=[D1]
[SUML]=[D0]
ΤΕΛΟΣ

```
        ORG $400400
N1H     DC.L $11111112
N1L     DC.L $EEEEEEEF
N2H     DC.L $EEEEEEEC
N2L     DC.L $11111113
SUM     DS.L 2

        ORG $400420
SBRTN   MOVE.L N1L,D0
        ADD.L N2L,D0
        MOVE.L D0,SUM+4
        MOVE.L N1H,D1
        MOVE.L N2H,D2
        ADDX.L D2,D1
        MOVE.L D1,SUM
        END SBRTN
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
 1  00400400                          ORG       $400400
 2  00400400 11111112      N1H:       DC.L      $11111112
 3  00400404 EEEEEEEF      N1L:       DC.L      $EEEEEEEF
 4  00400408 EEEEEEEC      N2H:       DC.L      $EEEEEEEC
 5  0040040C 11111113      N2L:       DC.L      $11111113
 6  00400410 00000008      SUM:       DS.L      2
 7
 8  00400420                          ORG       $400420
 9  00400420 203900400404  SUBRTN:    MOVE.L    N1L,D0
10  00400426 D0B90040040C              ADD.L     N2L,D0
11  0040042C 23C000400414              MOVE.L    D0,SUM+4
12  00400432 223900400400              MOVE.L    N1H,D1
13  00400438 243900400408              MOVE.L    N2H,D2
14  0040043E D382                      ADDX.L    D2,D1
15  00400440 23C100400410              MOVE.L    D1,SUM
16  00400420                          END       $400420
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 11 11 11 12 EE EE EE EF EE EE EE EC 11 11 11 13**

**400410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400420 20 39 00 40 04 04 D0 B9 00 40 04 0C 23 C0 00 40**

**400430 04 14 22 39 00 40 04 00 24 39 00 40 04 08 D3 82**

**400440 23 C1 00 40 04 10 00 00 00 00 00 00 00 00 00 00**

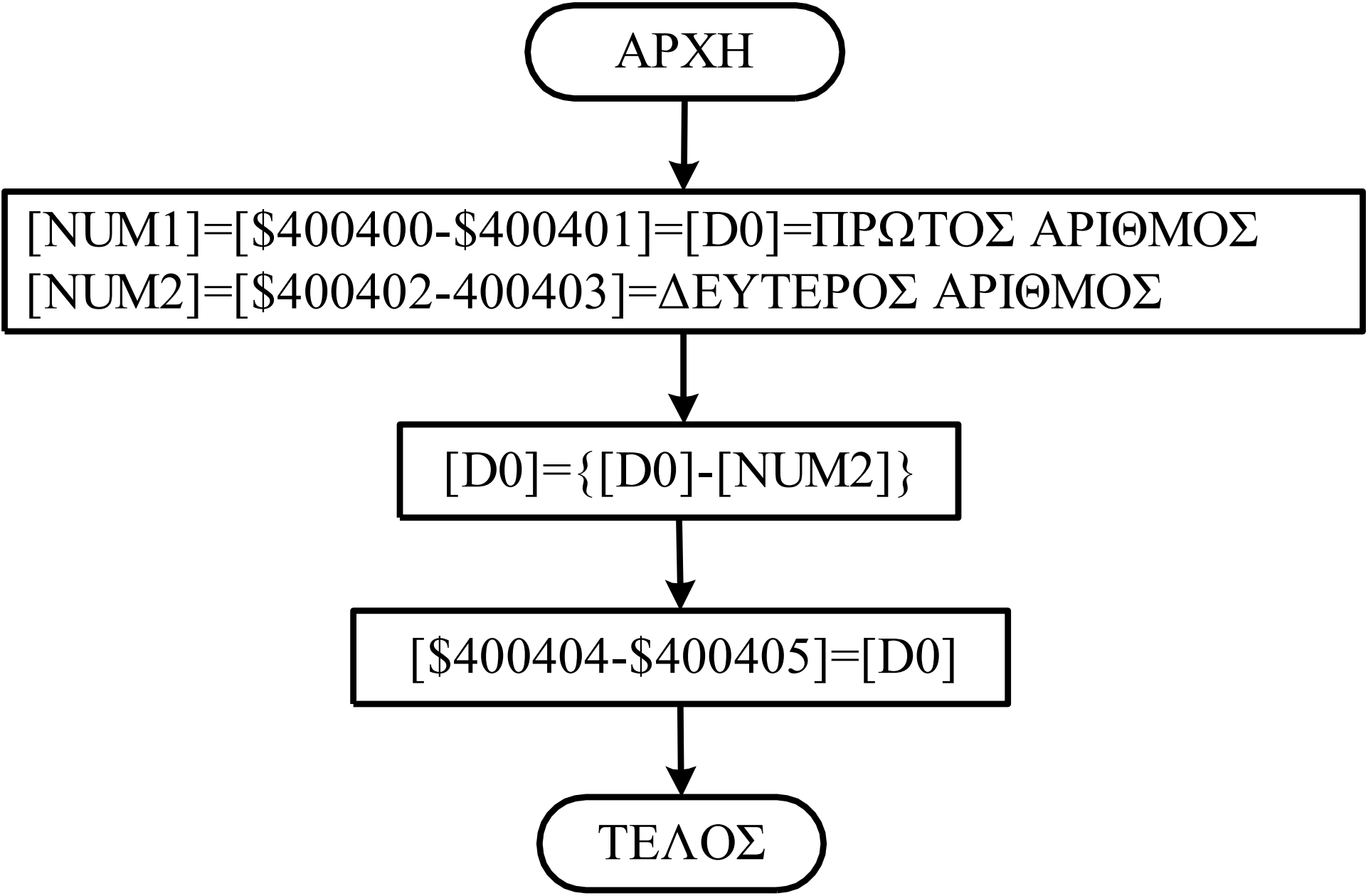
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 11 11 11 12 EE EE EE EF EE EE EE EC 11 11 11 13**

**400410 FF FF FF FF 00 00 00 02 00 00 00 00 00 00 00 00**

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που θα αφαιρεί από τον αριθμό μήκους λέξης που είναι αποθηκευμένος στη θέση μνήμης $400400 τον αριθμό μήκους λέξης που είναι αποθηκευμένος στη θέση 400402 και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400404.***


ΑΡΧΗ
[NUM1]=[$400400-$400401]=[D0]=ΠΡΩΤΟΣ ΑΡΙΘΜΟΣ
[NUM2]=[$400402-400403]=ΔΕΥΤΕΡΟΣ ΑΡΙΘΜΟΣ
[D0]={[D0]-[NUM2]}
[$400404-$400405]=[D0]
ΤΕΛΟΣ

```
                ORG $400400
NUM1            DC.W $5432
NUM2            DC.W $4321
DIF             DS.W  1

                ORG $400410
  SUBRTN        MOVE.W NUM1,D0
                SUB.W NUM2,D0
                MOVE.W D0,DIF
                RTS
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
1  00400400                          ORG       $400400
2  00400400 5432          NUM1:      DC.W      $5432
3  00400402 4321          NUM2:      DC.W      $4321
4  00400404 00000002      DIF:       DS.W      1
5
6  00400410                          ORG       $400410
7  00400410 303900400400  SUBRTN:    MOVE.W    NUM1,D0
8  00400416 907900400402             SUB.W     NUM2,D0
9  0040041C 33C000400404             MOVE.W    D0,DIF
10 00400410                          END       $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 54 32 43 21 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 30 39 00 40 04 00 90 79 00 40 04 02 33 C0 00 40**

**400420 04 04 00 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

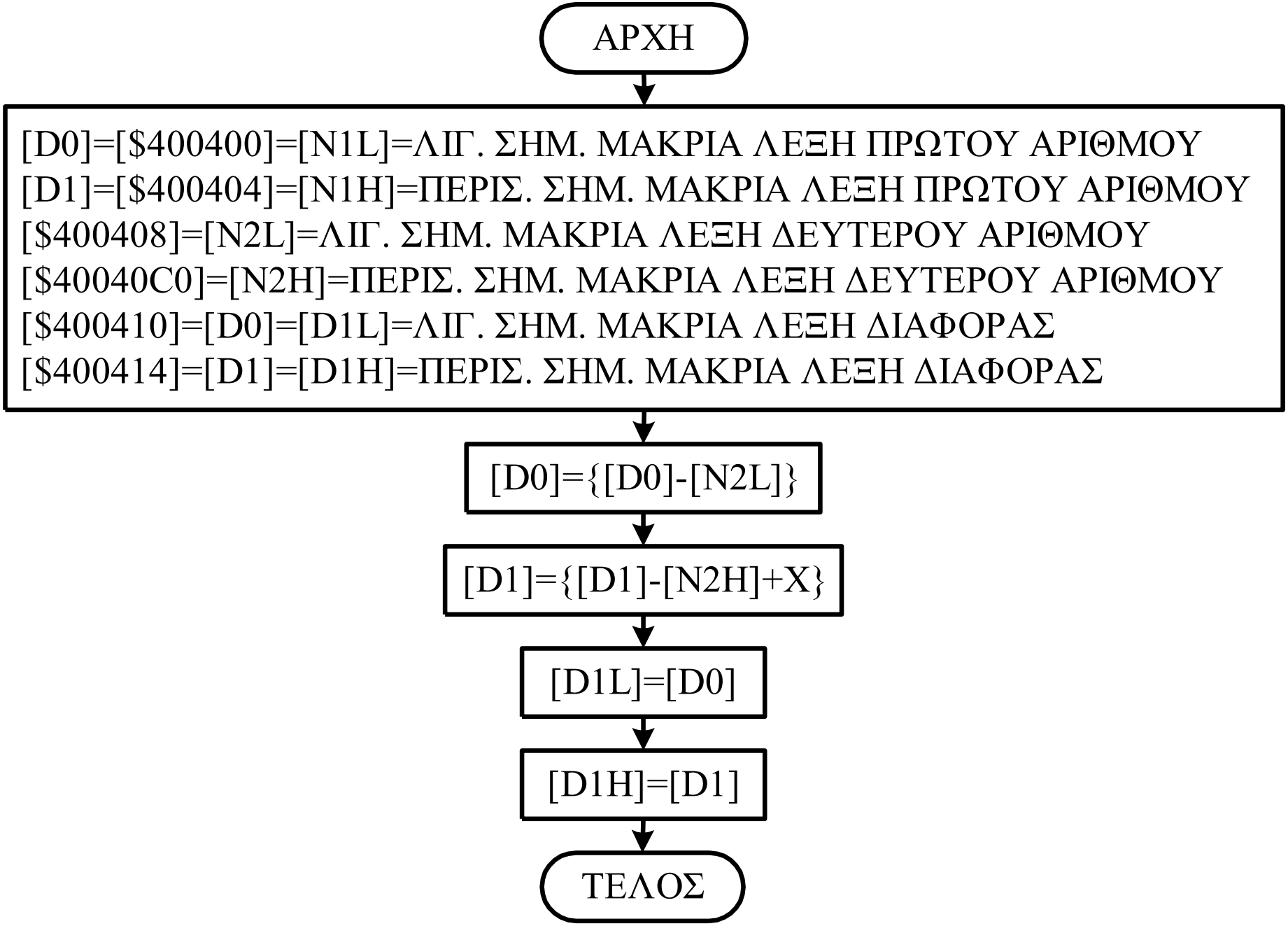
**400400 54 32 43 21 11 11 00 00 00 00 00 00 00 00 00**

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που θα κάνει αφαίρεση πολλαπλής ακρίβειας αφαιρώντας απ' τον αριθμό 64 ψηφίων (δύο μακριές λέξεις) που είναι αποθηκευμένος στις θέσεις μνήμης $400400-$400407, τον αριθμό 64 ψηφίων (δύο μακριές λέξεις) που είναι αποθηκευμένος στις θέσεις μνήμης $400408-$40040F.***

***Το αποτέλεσμα να αποθηκευτεί στις θέσεις μνήμης $400410-$400417.***

***(Η περισσότερο σημαντική μακριά λέξη βρίσκεται στις χαμηλότερες θέσεις μνήμης).***


ΑΡΧΗ
[D0]=[$400400]=[N1L]=ΛΙΓ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΠΡΩΤΟΥ ΑΡΙΘΜΟΥ
[D1]=[$400404]=[N1H]=ΠΕΡΙΣ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΠΡΩΤΟΥ ΑΡΙΘΜΟΥ
[$400408]=[N2L]=ΛΙΓ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΔΕΥΤΕΡΟΥ ΑΡΙΘΜΟΥ
[$40040C0]=[N2H]=ΠΕΡΙΣ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΔΕΥΤΕΡΟΥ ΑΡΙΘΜΟΥ
[$400410]=[D0]=[D1L]=ΛΙΓ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΔΙΑΦΟΡΑΣ
[$400414]=[D1]=[D1H]=ΠΕΡΙΣ. ΣΗΜ. ΜΑΚΡΙΑ ΛΕΞΗ ΔΙΑΦΟΡΑΣ
[D0]={[D0]-[N2L]}
[D1]={[D1]-[N2H]+X}
[D1L]=[D0]
[D1H]=[D1]
ΤΕΛΟΣ

```
        ORG $400400
N1H     DC.L $EEEEEEEC
N1L     DC.L $11111113
N2H     DC.L $11111112
N1L     DC.L $EEEEEEEF
DIF     DS.L 2

        ORG $400420
SBRTN   MOVE.L N1L,D0
        SUB.L N2L,D0
        MOVE.L D0,DIF+4
        MOVE.L N1H,D1
        SUBX.L N2H,D1
        MOVE.L D1,DIF
        RTS
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
1  00400400                          ORG       $400400
2  00400400 EEEEEEEC      N1H:       DC.L      $EEEEEEEC
3  00400404 11111113      N1L:       DC.L      $11111113
4  00400408 11111112      N2H        DC.L      $11111112
5  0040040C EEEEEEEF      N2L:       DC.L      $EEEEEEEF
6  00400410 00000008      DIF:       DS.L      2
7
8  00400420                          ORG       $400420
9  00400420 203900400404  SUBRTN:    MOVE.L    N1L,D0
10 00400426 90B90040040C             SUB.L     N2L,D0
11 0040042C 23C000400414             MOVE.L    D0,DIF+4
12 00400432 223900400400             MOVE.L    N1H,D1
13 00400438 243900400408             MOVE.L    N2H,D2
14 0040043E 9382                     SUBX.L    D2,D1
15 00400440 23C100400410             MOVE.L    D1,DIF
16 00400420                          END       $400420
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 EE EE EE EC 11 11 11 13 11 11 11 12 EE EE EE EF**

**400410 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400420 20 39 00 40 04 04 90 B9 00 40 04 0C 23 C0 00 40**

**400430 04 14 22 39 00 40 04 00 24 39 00 40 04 08 93 82**

**400440 23 C1 00 40 04 10 00 00 00 00 00 00 00 00 00 00**

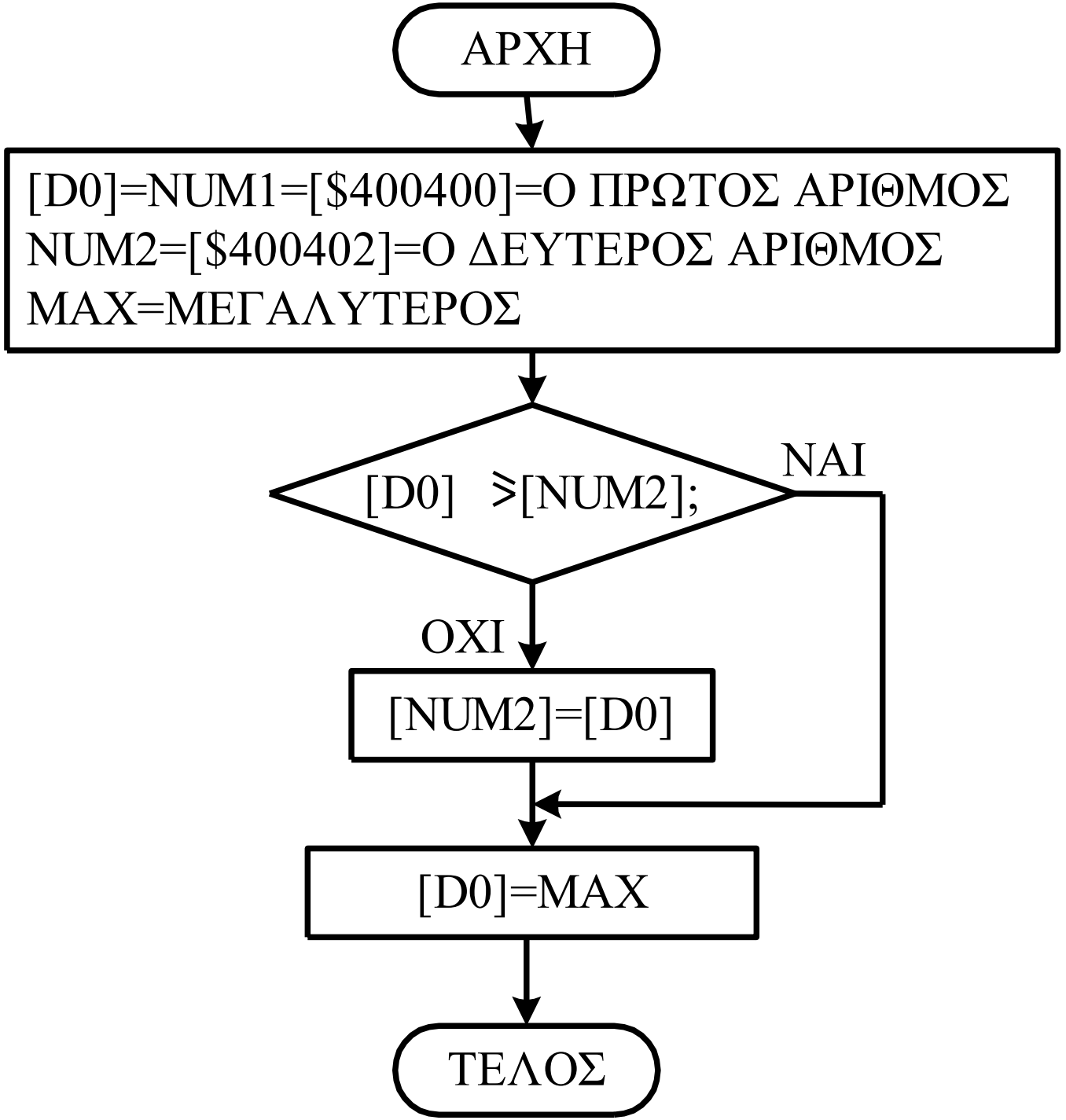
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 EE EE EE EC 11 11 11 13 11 11 11 12 EE EE EE EF**

**400410 DD DD DD D9 22 22 22 24 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.21

***Να γραφτεί μια υπορουτίνα που θα βρίσκει τη μεγαλύτερη τιμή από τους δύο αριθμούς μήκους λέξης (16 ψηφία) που είναι αποθηκευμένοι στις θέσεις μνήμης $400400 και $400402 αντίστοιχα, και θα τον αποθηκεύει στη θέση μνήμης $400404.***


ΑΡΧΗ
[D0]=NUM1=[$400400]=Ο ΠΡΩΤΟΣ ΑΡΙΘΜΟΣ
NUM2=[$400402]=Ο ΔΕΥΤΕΡΟΣ ΑΡΙΘΜΟΣ
MAX=ΜΕΓΑΛΥΤΕΡΟΣ
[D0] ⩾[NUM2];
ΝΑΙ
ΟΧΙ
[NUM2]=[D0]
[D0]=MAX
ΤΕΛΟΣ

```
        ORG    $400400
NUM1    DC.W   $2712
NUM2    DC.W   $355E
MAX     DS.W   1

        ORG        $400410
SUBRTN  MOVE.W NUM1,D0
        CMP.W    NUM2,D0
        BCC        DONE
        MOVE.W  NUM2,D0
DONE    MOVE.W  D0,MAX
        RTS
```

IF NUM1>NUM2
THEN C=0 -> TAKE BRANCH “DONE”,

IF NUM1<NUM2
THEN C=1 -> BRANCH NOT TAKEN.

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                          ORG      $400400
 2  00400400 2712            NUM1:    DC.W     $2712
 3  00400402 355E            NUM2:    DC.W     $355E
 4  00400404 00000002        MAX:     DS.W     1
 5
 6  00400410                          ORG      $400410
 7  00400410 303900400400    SUBRTN:  MOVE.W   NUM1,D0
 8  00400416 B07900400402             CMP.W    NUM2,D0
 9  0040041C 64000008                 BCC      DONE
10  00400420 303900400402             MOVE.W   NUM2,D0
11  00400426 33C000400404    DONE:    MOVE.W   D0,MAX
12 00400410                          END      $400410
```

α. Περιεχόμενο μνήμης δεδομένων πριν απ’ την εκτέλεση της υπορουτίνας.

**400400 27 12 35 5E 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιεχόμενο μνήμης κώδικα.

**400410 30 39 00 40 04 00 B0 79 00 40 04 02 64 00 00 08**

**400420 30 39 00 40 04 02 33 C0 00 40 04 04 00 00 00 00**

γ. Περιεχόμενο μνήμης δεδομένων μετά την εκτέλεση της υπορουτίνας.

**400400 27 12 35 5E 35 5E 00 00 00 00 00 00 00 00 00 00**

# MULS ΕΔ, Dn
# MULU ΕΔ, Dn

**Χρησιμοποιούνται όταν πρόκειται να γίνει πολλαπλασιασμός δύο προσημασμένων ή δύο μη προσημασμένων αριθμών αντιστοίχως.**

**Χρησιμοποιούν δύο τελεστέους 16 ψηφίων από τους οποίους ο ένας βρίσκεται στην ενεργό διεύθυνση και ο άλλος οπωσδήποτε σε έναν από τους καταχωρητές δεδομένων.**

**Το αποτέλεσμα είναι ένας αριθμός 32 ψηφίων που αποθηκεύεται στον καταχωρητή δεδομένων προορισμού.**

# MULS ΕΔ, Dn
# MULU ΕΔ, Dn

**Στην περίπτωση της εντολής MULS οι τελεστέοι θεωρούνται προσημασμένοι αριθμοί ενώ στην περίπτωση της εντολής MULU οι τελεστέοι θεωρούνται μη προσημασμένοι αριθμοί.**

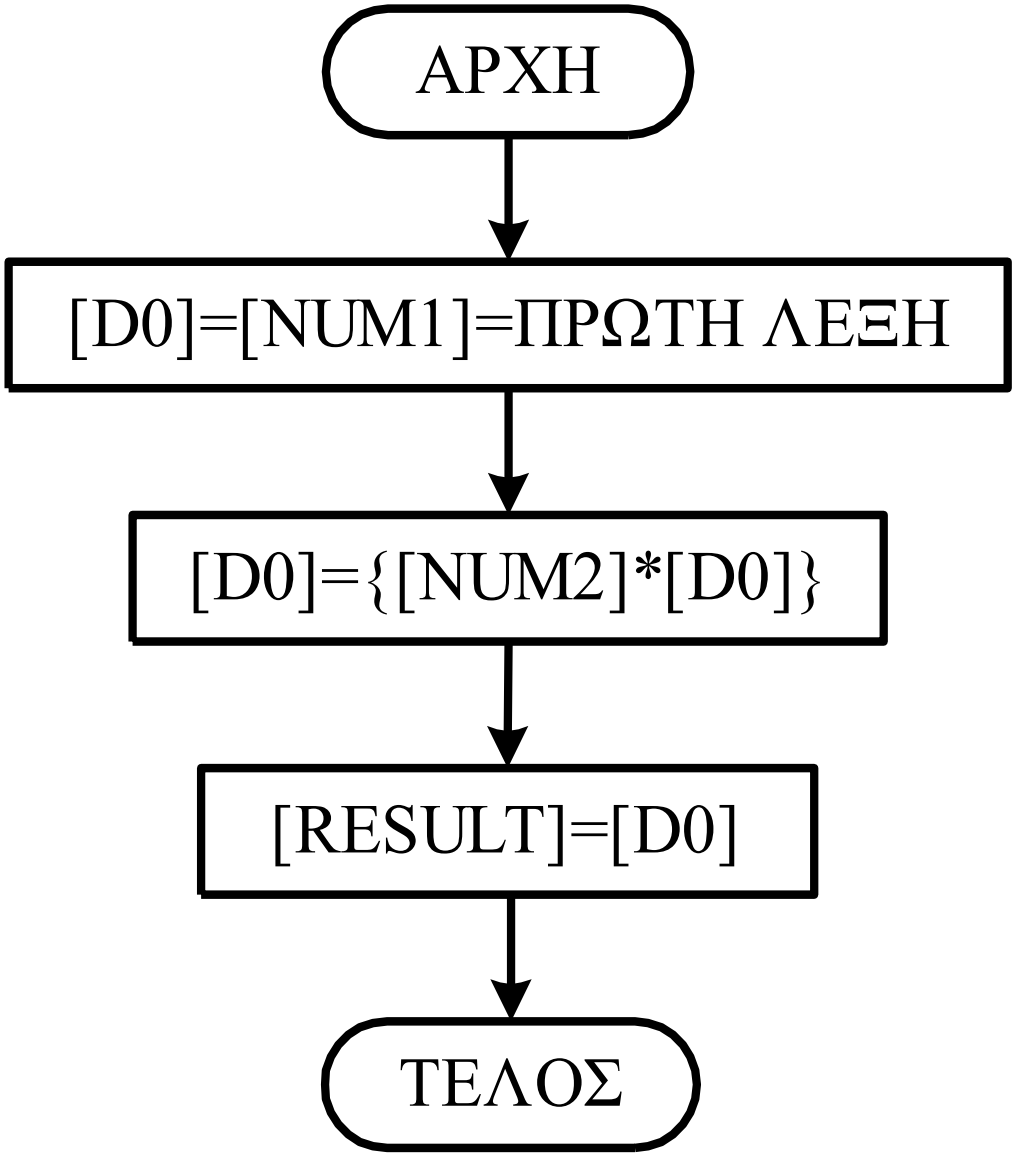
**Οι δείκτες του καταχωρητή κατάστασης που επηρεάζονται είναι οι N και Z που παίρνουν τιμές ανάλογα με το αποτέλεσμα και οι C και V που καθαρίζονται πάντα.**

# Παράδειγμα 3.10

***Να γραφτεί μια υπορουτίνα που θα πολλαπλασιάζει δύο μη-προσημασμένους αριθμούς μήκους λέξης που βρίσκονται στις θέσεις μνήμης $400400 και $400402 και θα αποθηκεύει τη μακριά λέξη του αποτελέσματος στη θέση μνήμης $400404.***

ΛΔ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

# Παράδειγμα 3.10




ΕΚΦΩΝΗΣΗ
ΠΡΟΓΡΑΜΜΑ ASSEMBLY
ΔΕΔΟΜΕΝΑ

```
                ORG $400400
NUM1            DC.W  $1111
NUM2            DC.W  $2000
PRODUCT         DS.L 1

                ORG $400410
SUBRTN          CLR.L D0
                MOVE.W NUM1,D0
                MULU NUM2,D0
                MOVE.L D0,PRODUCT
                RTS
```

ΕΚΦΩΝΗΣΗ
ΛΔ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                          ORG       $400400
2  00400400 1111          NUM1:      DC.W      $1111
3  00400402 2000          NUM2:      DC.W      $2000
4  00400404 00000004      PRODUCT:   DS.L      1
5
6  00400410                          ORG       $400410
7  00400410 4280          SUBRTN:    CLR.L     D0
8  00400412 303900400400             MOVE.W    NUM1,D0
9  00400418 C0F900400402             MULU      NUM2,D0
10  0040041E 23C000400404            MOVE.L    D0,PRODUCT
11  00400410                         END       $400410
```

ΕΚΦΩΝΗΣΗ
ΛΔ
ΠΡΟΓΡΑΜΜΑ
ΔΕΔΟΜΕΝΑ

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 11 11 20 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 42 80 30 39 00 40 04 00 C0 F9 00 40 04 02 23 C0**

**400420 00 40 04 04 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 11 11 20 00 02 22 20 00 00 00 00 00 00 00 00 00**

ΕΚΦΩΝΗΣΗ
ΛΔ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY

# DIVS ΕΔ, Dn
# DIVU ΕΔ, Dn

**Χρησιμοποιούνται όταν πρόκειται να γίνει διαίρεση δύο προσημασμένων ή δύο μη προσημασμένων αριθμών αντιστοίχως.**

**Ο τελεστέος προορισμού, που είναι ο διαιρετέος και είναι μια ψηφιολέξη μήκους 32 ψηφίων, είναι πάντοτε ένας από τους καταχωρητές δεδομένων του μικροεπεξεργαστή ενώ ο τελεστέος προέλευσης, που είναι ο διαιρέτης και είναι μια ψηφιολέξη 16 ψηφίων, μπορεί να προσπελαστεί χρησιμοποιώντας μια οποιαδήποτε από τις μεθόδους του μικροεπεξεργαστή M68000.**

# DIVS ΕΔ, Dn
# DIVU ΕΔ, Dn

**Το πηλίκο της διαίρεσης αποθηκεύεται στα 16 λιγότερο σημαντικά ψηφία του τελεστέου προορισμού και το υπόλοιπο στα 16 περισσότερο σημαντικά ψηφία του τελεστέου προορισμού.**

**Το πρόσημο του αποτελέσματος που προέρχεται από μια προσημασμένη διαίρεση είναι πάντοτε το ίδιο με το πρόσημο του διαιρετέου.**

**Οι δείκτες του καταχωρητή κατάστασης που επηρεάζονται είναι οι N και Z που παίρνουν τιμές ανάλογα με το αποτέλεσμα. Αν το αποτέλεσμα της διαίρεσης είναι αριθμός με περισσότερα από 16 ψηφία τότε ο δείκτης υπερχείλισης παίρνει την τιμή "1".**

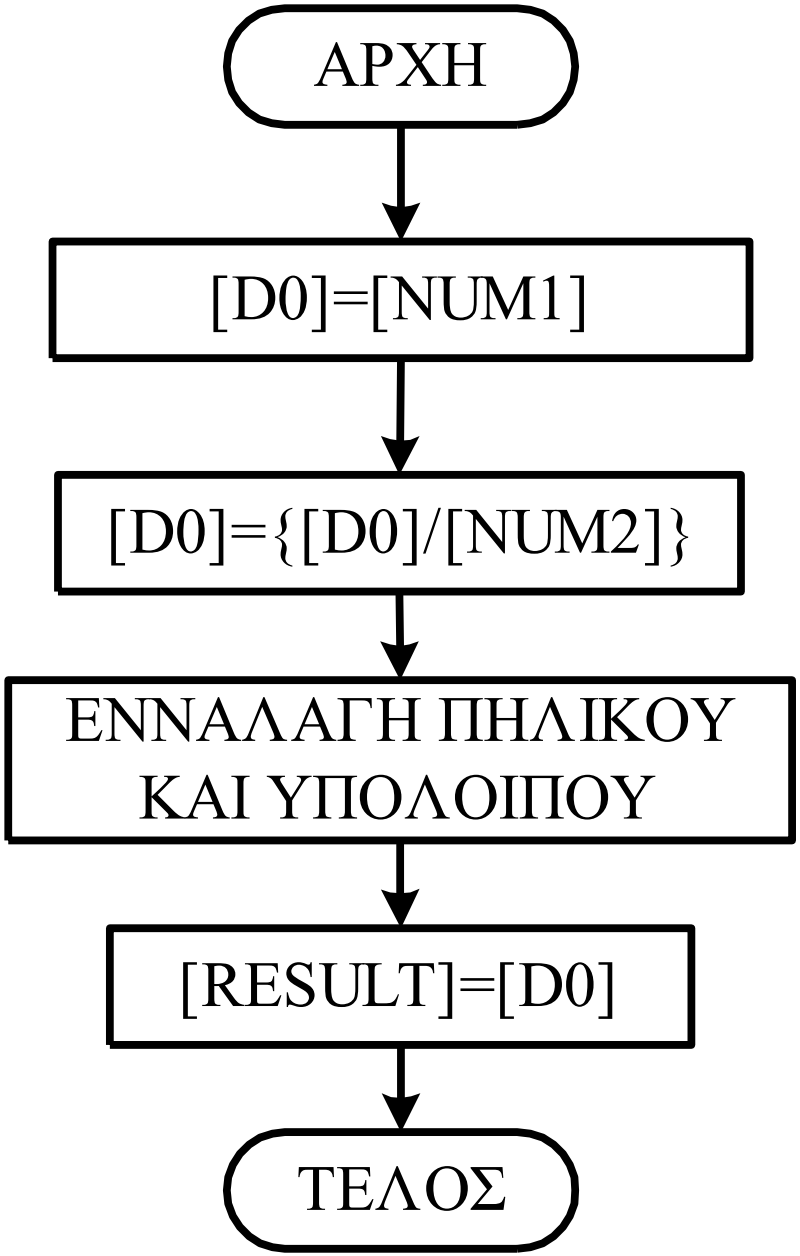
# Παράδειγμα 3.11

***Να γραφτεί μια υπορουτίνα που θα διαιρεί το μη-προσημασμένο αριθμό μήκους μακριάς λέξης που βρίσκεται στη θέση μνήμης $400400 με το μη-προσημασμένο αριθμό μήκους λέξης που βρίσκεται στη θέση μνήμης $400404.***

***Το πηλίκο να αποθηκευτεί στη θέση μνήμης $400406 και το υπόλοιπο στη θέση μνήμης $400408.***

ΛΔ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

# Παράδειγμα 3.11




ΕΚΦΩΝΗΣΗ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

```
                ORG $400400
NUM1            DC.L  15
NUM2            DC.W  7
RESULT          DS.L 1

                ORG $400410
SUBRTN          CLR.L D0
                MOVE.L NUM1,D0
                DIVU NUM2,D0
                SWAP D0
                MOVE.L D0,RESULT
                RTS
```

ΕΚΦΩΝΗΣΗ
ΛΔ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                          ORG       $400400
 2  00400400 0000000F      NUM1:      DC.L      15
 3  00400404 0007          NUM2:      DC.W      7
 4  00400406 00000004      RESULT:    DS.L      1
 5
 6  00400410                          ORG       $400410
 7  00400410 4280          SUBRTN:    CLR.L     D0
 8  00400412 203900400400             MOVE.L    NUM1,D0
 9  00400418 80F900400404             DIVU      NUM2,D0
10  0040041E 4840                     SWAP      D0
11  00400420 23C000400406             MOVE.L    D0,RESULT
12 00400410                           END       $400410
```

ΕΚΦΩΝΗΣΗ
ΛΔ
ΠΡΟΓΡΑΜΜΑ
ΔΕΔΟΜΕΝΑ

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 00 00 00 0F 00 07 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 42 80 20 39 00 40 04 00 80 F9 00 40 04 04 48 40**

**400420 23 C0 00 40 04 06 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 00 00 00 0F 00 07 00 02 00 01 00 00 00 00 00 00**

ΕΚΦΩΝΗΣΗ
ΛΔ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY

# EXT.L D0

**SING EXTEND**

**Πριν την εκτέλεση της εντολής**

**[D0]=$0000F100**

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000000000000001111000100000000**

**Μετά την εκτέλεση της εντολής**

**[D0]=$FFFFF100**

**Μετά την εκτέλεση της εντολής**

**[D0]=$11111111111111111111000100000000**

# Παράδειγμα 3.12

***Υποτίθεται ότι οι καταχωρητές D0, D1 και D2 περιέχουν ένα προσημασμένο byte, μια προσημασμένη λέξη και μια προσημασμένη μακριά λέξη σε μορφή συμπληρώματος ¨ως προς 2¨, αντίστοιχα.***

***Να γραφεί μια υπορουτίνα που θα δίνει το προσημασμένο αποτέλεσμα της πράξης D0=D0+D1-D2.***

****Παρατήρηση:** Πριν εκτελεστεί το πρόγραμμα πρέπει να μετατραπούν το byte και η λέξη των D0 και D1 αντίστοιχα σε προσημασμένες μακριές λέξεις. Το byte του D0 πρέπει να μετατραπεί πρώτα σε λέξη και στη συνέχεια σε μακριά λέξη.*

ΠΡΟΓΡΑΜΜΑ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

```
            ORG $400400
NUM1        DC.L $7FFF0000
NUM2        DC.W $7F00
NUM3        DC.B $FE
EMPTY       DS.B 1
RSLT        DS.L 1


            ORG $400410
SUBRTN      MOVE.L NUM1,D0
            MOVE.W NUM2,D1
            MOVE.B NUM3,D2
            EXT.W D2
            EXT.L D2
            EXT.L D1
            ADD.L D1,D0
            SUB.L D2,D0
            MOVE.L D0,RSLT
            RTS
```

ΕΚΦΩΝΗΣΗ
ASSEMBLY
ΔΕΔΟΜΕΝΑ

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                          ORG      $400400
2  00400400 7FFF0000       NUM1:     DC.L     $7FFF0000
3  00400404 7F00           NUM2:     DC.W     $7F00
4  00400406 FE             NUM3:     DC.B     $FE
5  00400407 00000001       EMPTY:    DS.B      1
6  00400408 00000004       RSLT:     DS.L      1
7
8  00400410                          ORG      $400410
9  00400410 203900400400   SUBRTN:   MOVE.L   NUM1,D0
10 00400416 323900400404             MOVE.W   NUM2,D1
11 0040041C 143900400406             MOVE.B   NUM3,D2
12 00400422 4882                     EXT.W    D2
13 00400424 48C2                     EXT.L    D2
14 00400426 48C1                     EXT.L    D1
15 00400428 D081                     ADD.L    D1,D0
16 0040042A 9082                     SUB.L    D2,D0
17 0040042C 23C000400408             MOVE.L   D0,RSLT
18 00400410                          END      $400410
```

ΕΚΦΩΝΗΣΗ
ΠΡΟΓΡΑΜΜΑ
ΔΕΔΟΜΕΝΑ

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 7F FF 00 00 7F 00 FE 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 20 39 00 40 04 00 32 39 00 40 04 04 14 39 00 40**

**400420 04 06 48 82 48 C2 48 C1 D0 81 90 82 23 C0 00 40**

**400430 04 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 7F FF 00 00 7F 00 FE 00 7F FF 7F 02 00 00 00 00**

ΕΚΦΩΝΗΣΗ
ΠΡΟΓΡΑΜΜΑ
ASSEMBLY

# ABCD D2,D3

## ADD DECIMAL WITH EXTEND

**Πριν την εκτέλεση της εντολής** **Size = byte**

**[D2]=$01000101_{BCD}$=$45_{10}$, [D3]=$00010011_{BCD}$=$13_{10}$, X=1**

**Μετά την εκτέλεση της εντολής**

**[D3]=$01011001_{BCD}$=$59_{10}$**

**[D3]=[D3]+[D2]+X**

**=$00010011_{BCD}$ +$01000101_{BCD}$ +1=$01011001_{BCD}$**

ADD DECIMAL WITH EXTEND

Syntax: ABCD Dy,Dx or ABCD -(Ay),-(Ax) Size = byte

# Ο κώδικας BCD

Ο BCD πήρε το όνομά του από τα ακρωνύμια των λέξεων Binary Coded Decimal (δυαδικά κωδικοποιημένος δεκαδικός) και οι αριθμοί 8421 συμβολίζουν τα "βάρη" των δυαδικών ψηφίων στην αντίστοιχη στήλη.

Δηλαδή ο αριθμός $0110_2$ είναι $0x8+1x4+1x2+0x1=6_{10}$

Λόγω της ύπαρξης των βαρών ο κώδικας ονομάζεται ζυγοσταθμισμένος.

Ο BCD κάνει δυαδική μετατροπή αριθμών από το 0 έως και το 9.

# Ο κώδικας BCD

Για παράδειγμα, ο δεκαδικός 67 είναι ο δυαδικός 1000011, ενώ ο αντίστοιχος δυαδικά-κωδικοποιημένος είναι ο 0110 0111, και ο

$375_{10}=1011101111_2=0011\ 0111\ 0101_{BCD8421}$

***Το βασικό που πρέπει να κατανοηθεί είναι, ότι ένας αριθμός BCD δεν είναι δυαδικός αλλά δυαδικά-κωδικοποιημένος αριθμός.***

# Πρόσθεση αριθμών BCD

***Έστω, για παράδειγμα, ότι πρέπει να προστεθούν οι αριθμοί***

***$52_{10}$= 0101 $0010_{8421BCD}$ και $37_{10}$= 0011 $0111_{8421BCD}$***

$$\begin{array}{r} 52_{10} = 0101\ 0010_{8421BCD} \\ +37_{10} = 0011\ 0111_{8421BCD} \\ \hline 89_{10} = 1000\ 1001_{8421BCD} \end{array}$$

**Παρατηρείται ότι, προστίθενται τα δύο λιγότερο σημαντικά ψηφία και, αν δεν προκύψει κρατούμενο, τα δύο περισσότερο σημαντικά ψηφία των δύο αριθμών.**

**Όταν προκύπτει κρατούμενο από την πρόσθεση των λιγότερο σημαντικών ψηφίων τότε αυτό πρέπει να προστεθεί στη στήλη των επόμενων σε σημαντικότητα ψηφίων.**

**Κρατούμενο προκύπτει όταν το άθροισμα των δύο ψηφίων είναι αριθμός μεγαλύτερος του $1001_2=9_{10}$. Δηλαδή 10, 11, 12, 13, 14, 15.**

**Στην περίπτωση αυτή θα πρέπει στο άθροισμα που προκύπτει να προστεθεί ο αριθμός έξι, όσος και ο αριθμός των δυαδικών συνδυασμών που δε χρησιμοποιούνται για την παράσταση των αριθμών BCD. Η διαδικασία αυτή λέγεται *'διόρθωση κρατούμενου'* και φαίνεται στο παράδειγμα που ακολουθεί.**

*Ας υποτεθεί ότι πρόκειται να προστεθούν οι αριθμοί, $427_{10}$= 0100 0010 $0111_{8421BCD}$ και $238_{10}$= 0010 0011 $1000_{8421BCD}$*

**$427_{10}$= 0100 0010 $0111_{8421BCD}$**

**+ $238_{10}$= 0010 0011 $1000_{8421BCD}$**

**1111 ⇒ άκυρο άθροισμα**

**Κρατούμενο ⇒ 1 0110 ⇒ πρόσθεση 6 για διόρθωση κρατούμενου**

**= $665_{10}$= 0110 0110 $0101_{8421BCD}$**

**Παρατηρείται εδώ ότι τα δύο λιγότερο σημαντικά ψηφία θα δώσουν άθροισμα 15 που είναι αριθμός μεγαλύτερος του $9_{10}$. Όταν στα ψηφία αυτά προστίθεται το έξι τότε προκύπτει ο αριθμός $15_{10}$ (άθροισμα 5 και κρατούμενο 1). Το ψηφίο 5 θα παραμείνει ως το λιγότερο σημαντικό ψηφίο του αθροίσματος και το κρατούμενο θα προστεθεί στα επόμενα περισσότερο σημαντικά ψηφία των αριθμών για να δώσει άθροισμα 6 κ.ο.κ.**

*Ας υποτεθεί ότι πρόκειται να προστεθούν οι αριθμοί, $429_{10}$= 0100 0010 $1001_{8421BCD}$ και $238_{10}$= 0010 0011 $1000_{8421BCD}$.*

**$429_{10}$= 0100 0010 $1001_{8421BCD}$**
**+$238_{10}$= 0010 0011 $1000_{8421BCD}$**
**Κρατούμενο ⇒1 0001 ⇒ άκυρο άθροισμα**
**0110 ⇒ πρόσθεση 6 για διόρθωση κρατούμενου**
**= $667_{10}$= 0110 0110 $0111_{8421BCD}$**

**Εδώ παρατηρείται ότι τα δύο λιγότερο σημαντικά ψηφία δίνουν άθροισμα 17 (μεγαλύτερο του 15). Αυτό σημαίνει ότι το κρατούμενο που θα προστεθεί ήδη έχει παραχθεί επομένως ο αριθμός έξι προστίθεται μόνο για να γίνει η σχετική διόρθωση.**

# ABCD -(A0),-(A1)

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400502, [A1]=$00400602**

**[$400501]=$00100111_{BCD}=27_{10}$, [$400601]=$00111001_{BCD}=39_{10}$**

**και X=1**

**Μετά την εκτέλεση της εντολής**

**[$400601]=$01100111_{BCD}=67_{10}$**

**[$400601]=[$400601]+[$400501]+X**

**$= 00111001_{BCD} + 00100111_{BCD} + 1 = 01100111_{BCD}$**

# SBCD D3,D2

**Πριν την εκτέλεση της εντολής**

**[D2]=$01000101_{BCD}=45_{10}$, [D3]=$00010011_{BCD}=13_{10}$, X=1**

**Μετά την εκτέλεση της εντολής**

**[D2]=$00110001_{BCD}=31_{10}$**

**[D2]=[D2]-[D3]-X**

**=$01000101_{BCD}$ -$00010011_{BCD}$ -1=$00110001_{BCD}$**

# SBCD -(A0),-(A1)

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400502, [A1]=$00400602**

**[$400501]=$00100111_{BCD}=27_{10}$, [$400601]=$00111001_{BCD}=39_{10}$**

**και X=1**

**Μετά την εκτέλεση της εντολής**

**[$400601]=$00010001_{BCD}=11_{10}$**

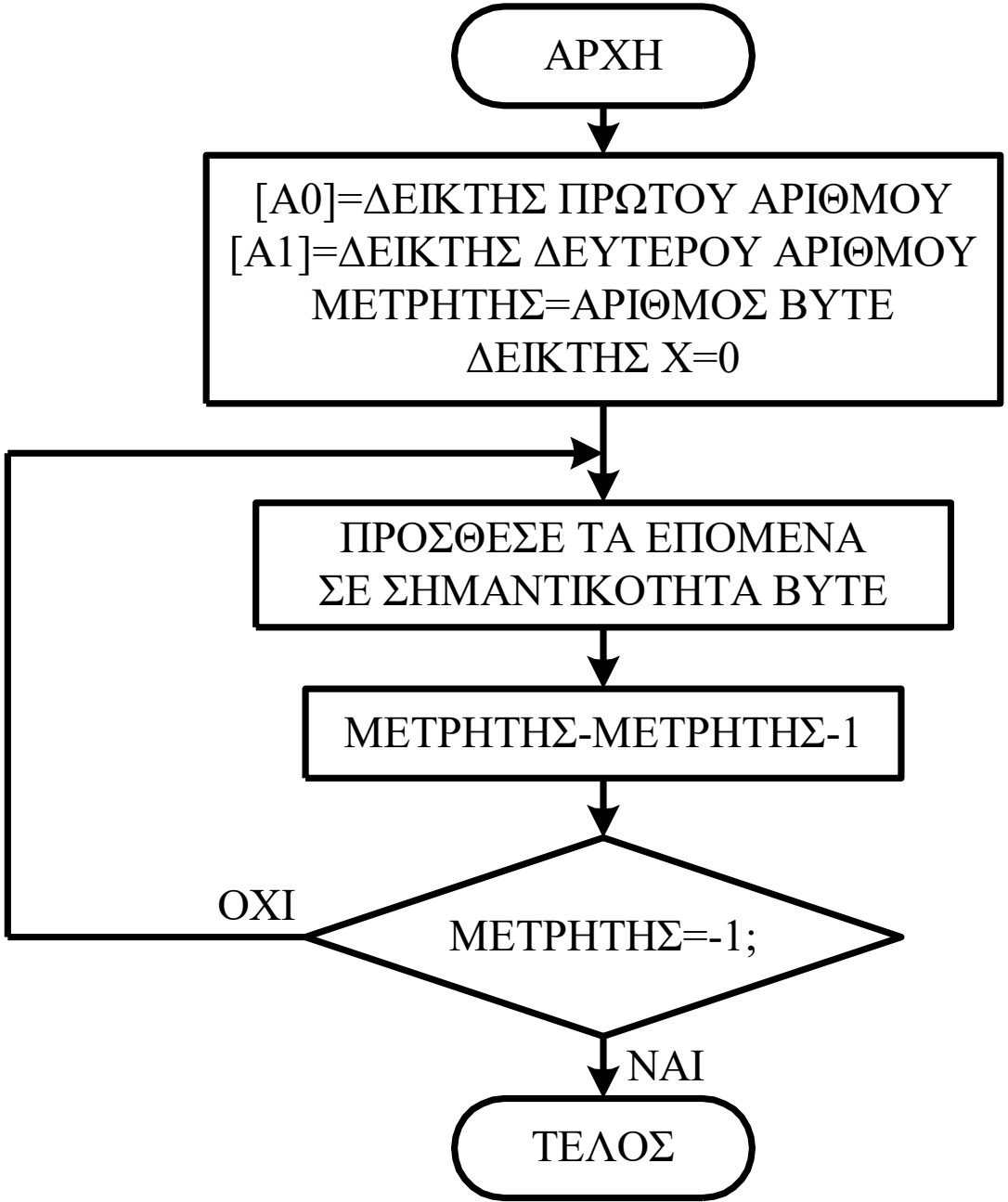
**[$400601]=[$400601]-[$400501]-X**

**$= 00111001_{BCD} - 00100111_{BCD} - 1 = 00010001_{BCD}$**

# Παράδειγμα 3.13

***Να γραφτεί μια υπορουτίνα που θα προσθέτει δύο BCD αριθμούς οκτώ ψηφίων (4 byte) που είναι αποθηκευμένοι στις θέσεις μνήμης $400400-400403 και $400404-400407 και να αποθηκεύει το αποτέλεσμα στις θέσεις μνήμης 400404-40007.***

# Παράδειγμα 3.13



```
        ORG $400400
NUM1    DC.L $11443326
NUM2    DC.L $87655812
        ORG $400410
ADDBCD  LEA NUM1+4,A0
        LEA NUM2+4,A1
        MOVE #$04,CCR
        MOVE.B #03,D0
LOOP    ABCD -(A0),-(A1)
        DBRA D0,LOOP
        RTS
```

Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
 1  00400400                          ORG     $400400
 2  00400400 11443326      NUM1:      DC.L    $11443326
 3  00400404 87655812      NUM2:      DC.L    $87655812
 4  00400410                          ORG     $400410
 5  00400410 41F900400404  ADDBCD:    LEA     NUM1+4,A0
 6  00400416 43F900400408              LEA     NUM2+4,A1
 7  0040041C 44FC0004                  MOVE    #04,CCR
 8  00400420 103C0003                  MOVE.B  #03,D0
 9  00400424 C308          LOOP:       ABCD    -(A0),-(A1)
10  00400426 51C8FFFC                  DBRA    D0,LOOP
11  00400410                           END     $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 11 44 33 26 87 65 58 12 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 41 F9 00 40 04 04 43 F9 00 40 04 08 44 FC 00 04**

**400420 10 3C 00 03 C3 08 51 C8 FF FC**

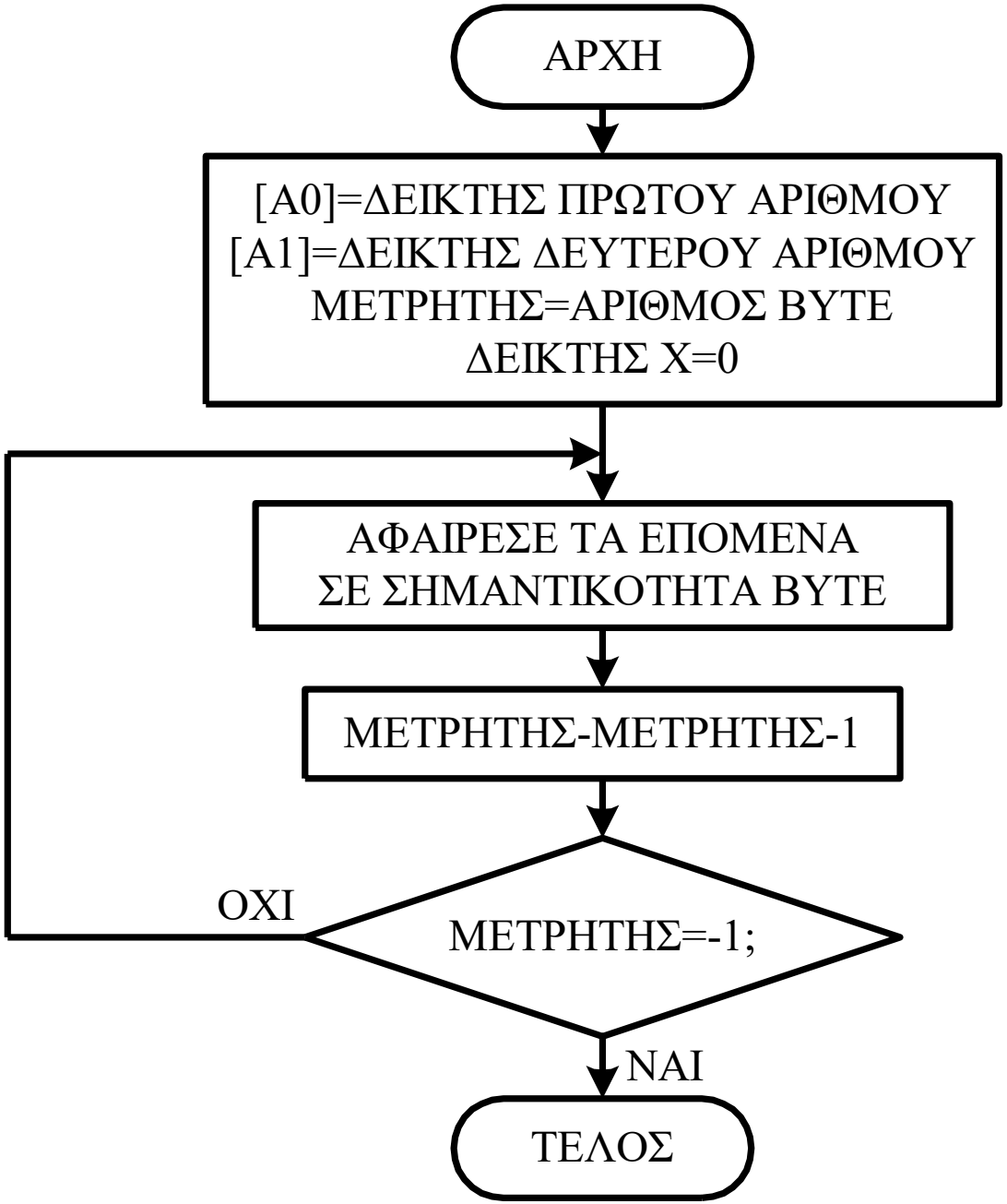
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 11 44 33 26 99 09 91 38 00 00 00 00 00 00 00 00**

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που θα αφαιρεί απ' το BCD αριθμό οκτώ ψηφίων (4 byte) που είναι αποθηκευμένος στις θέσεις μνήμης $400400 - 400403 το BCD αριθμό οκτώ ψηφίων (4 byte) που είναι αποθηκευμένος στις θέσεις μνήμης $400404 - 400407 και να αποθηκεύει το αποτέλεσμα στις θέσεις μνήμης 400400 - 400403.***

# Παράδειγμα


ΑΡΧΗ
[Α0]=ΔΕΙΚΤΗΣ ΠΡΩΤΟΥ ΑΡΙΘΜΟΥ
[Α1]=ΔΕΙΚΤΗΣ ΔΕΥΤΕΡΟΥ ΑΡΙΘΜΟΥ
ΜΕΤΡΗΤΗΣ=ΑΡΙΘΜΟΣ ΒΥΤΕ
ΔΕΙΚΤΗΣ Χ=0
ΑΦΑΙΡΕΣΕ ΤΑ ΕΠΟΜΕΝΑ
ΣΕ ΣΗΜΑΝΤΙΚΟΤΗΤΑ ΒΥΤΕ
ΜΕΤΡΗΤΗΣ-ΜΕΤΡΗΤΗΣ-1
ΟΧΙ
ΜΕΤΡΗΤΗΣ=-1;
ΝΑΙ
ΤΕΛΟΣ

```
        ORG $400400
NUM1    DC.L $87655812
NUM2    DC.L $11443326
        ORG $400410
SUBBCD  LEA NUM1+4,A0
        LEA NUM2+4,A1
        MOVE #$04,CCR
        MOVE.B #03,D0
LOOP    SBCD -(A0),-(A1)
        DBRA D0,LOOP
        RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής**

**θα δώσει την παρακάτω λίστα:**

```
1  00400400                              ORG     $400400
2  00400400 87655812     NUM1:           DC.L    $87655812
3  00400404 11443326     NUM2:           DC.L    $11443326
4  00400410                              ORG     $400410
5  00400410 41F900400404 SUBBCD:         LEA     NUM1+4,A0
6  00400416 43F900400408                 LEA     NUM2+4,A1
7  0040041C 44FC0004                     MOVE    #$04,CCR
8  00400420 103C0003                     MOVE.B  #03,D0
9  00400424 8308         LOOP:           SBCD    -(A0),-(A1)
10 00400426 51C8FFC                      DBRA    D0,LOOP
11  00400410                             END      $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:
**400400 87 65 58 12 11 44 33 26 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα
**400410 41 F9 00 40 04 04 43 F9 00 40 04 08 44 FC 00 04**
**400420 10 3C 00 03 83 08 51 C8 FF C0 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:
**400400 76 21 24 86 11 44 33 26 00 00 00 00 00 00 00 00**

# Λογικές Εντολές

**AND για να καθαρίζουμε θέσεις Bit στο 0 (clear)**
**OR για να τα θέτουμε στον 1 (set)**
**και EOR για να τα αλλάξουμε τιμή (toggle)**

# ANDI.B #$4B,D0

No Support of An

**Πριν την εκτέλεση της εντολής**

**[D0]=$00FFFFFF**

**Μετά την εκτέλεση της εντολής**

**[D0]=$00FFFF4B**

**Λιγότερο Σημαντικό byte του D0 = $FF =1111 1111**

**AND $4B = 0100 1011**

**Αποτέλεσμα = 0100 1011=$4B**

# ANDI.B #$EB,CCR

**Η εντολή αυτή θα κάνει λογική πράξη AND των περιεχομένων του κώδικα συνθήκης του καταχωρητή κατάστασης με τον αριθμό $EB=$11101011_2$**

**Μετά την εκτέλεση της εντολής θα καθαριστούν οι δείκτες X και Z και θα μείνουν ανεπηρέαστοι οι δείκτες N, V και C.**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | 0 | N | 0 | V | C |
| T | | S | | | I2 | I1 | I0 | | | X | N | Z | V | C |
| | | | | | | | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |

# AND.L D0,D7

**Πριν την εκτέλεση της εντολής**

**[D0]=$0000FFFF, [D7]=$1F33AC12**

**Μετά την εκτέλεση της εντολής**

**[D7]=$0000AC12**

**$0000FFFF = 0000 0000 0000 0000 1111 1111 1111 1111**

**$1F33AC12 = 0001 1111 0011 0011 1010 1100 0001 0010**

**$0000AC12 = 0000 0000 0000 0000 1010 1100 0001 0010**

# OR.L (A0),D0

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400500, [$400500]=$13FE350D, [D0]=$AB3517B0**

**Μετά την εκτέλεση της εντολής**

**[D0]=$BBFF37BD**

**$13FE350D = 0001 0011 1111 1110 0011 0101 0000 1101**

**$AB3517B0 = 1010 1011 0011 0101 0001 0111 1011 0000**

**$BBFF37BD =1011 1011 1111 1111 0011 0111 1011 1101**

# ORI.B #$15,CCR

**Η εντολή αυτή θα κάνει λογική πράξη OR των περιεχομένων του κώδικα συνθήκης του καταχωρητή κατάστασης με τον αριθμό $15=00010101$_2$**

**Μετά την εκτέλεση της εντολής οι δείκτες X, Z και C θα γίνουν ¨1¨ και θα μείνουν ανεπηρέαστοι οι δείκτες N, V.**

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | | 1 | N | 1 | V | 1 |
| T | | S | | | I2 | I1 | I0 | | | X | N | Z | V | C |
| | | | | | | | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |

# Παράδειγμα 3.14

***Να βρεθεί η μάσκα που θα κάνει τη λογική πράξη AND και θα μηδενίζει τα τέσσερα λιγότερο σημαντικά ψηφία του byte δεδομένων, που είναι αποθηκευμένο στη θέση μνήμης $400400, αφήνοντας ανεπηρέαστα τα τέσσερα περισσότερο σημαντικά. Επίσης θα καθαρίζει τα τέσσερα λιγότερο σημαντικά ψηφία του καταχωρητή D0 και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400404.***

```
                ORG $400400
NUM             DC.L  $12345678
RESULT          DS.L 1

                ORG $400410
SUBRTN          MOVE.L NUM,D0
                ANDI.B #$F0,NUM
                ANDI.L #$FFFFFFF0,D0
                MOVE.L D0,RESULT
                RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                          ORG       $400400
2  00400400 12345678        NUM:     DC.L      $12345678
3  00400404 00000004        RESULT:  DS.L      1
4
5  00400410                          ORG       $400410
6  00400410 203900400400    SUBRTN:  MOVE.L    NUM,D0
7  00400416 023900F000400400         ANDI.B    #$F0,NUM
8  0040041E 0280FFFFFFF0             ANDI.L    #$FFFFFFF0,D0
9  00400424 23C000400404             MOVE.L    D0,RESULT
10 00400410                          END       $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 12 34 56 78 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 20 39 00 40 04 00 02 39 00 F0 00 40 04 00 02 80**

**400420 FF FF FF F0 23 C0 00 40 04 04 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 10 34 56 78 12 34 56 70 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.15

***Να βρεθεί η μάσκα που θα κάνει τη λογική πράξη OR και θα κάνει ¨1¨ τα τέσσερα λιγότερο σημαντικά ψηφία του byte δεδομένων, που είναι αποθηκευμένο στη θέση μνήμης $400400, αφήνοντας ανεπηρέαστα τα τέσσερα περισσότερο σημαντικά. Επίσης θα κάνει ¨1¨ τα τέσσερα λιγότερο σημαντικά ψηφία του καταχωρητή D0 και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400404.***

```
                ORG  $400400
NUM             DC.L  $12345678
RESULT          DS.L   1


                ORG       $400410
SUBRTN          MOVE.L  NUM,D0
                ORI.B     #$0F,NUM
                ORI.L      #$0000000F,D0
                MOVE.L   D0,RESULT
                RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                          ORG       $400400
2  00400400 12345678      NUM:       DC.L      $12345678
3  00400404 00000004      RESULT:    DS.L      1
4
5  00400410                          ORG       $400410
6  00400410 203900400400  SUBRTN:    MOVE.L    NUM,D0
7  00400416 0039000F00400400         ORI.B     #$0F,NUM
8  0040041E 00800000000F             ORI.L     #$0000000F,D0
9  00400424 23C000400404             MOVE.L    D0,RESULT
10 00400410                          END       $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 12 34 56 78 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 20 39 00 40 04 00 00 39 00 0F 00 40 04 00 00 80**

**400420 00 00 00 0F 23 C0 00 40 04 04 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 1F 34 56 78 12 34 56 7F 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.16

***Να βρεθεί η μάσκα που θα κάνει τη λογική πράξη EXOR και θα αντιστρέφει τα τέσσερα λιγότερο σημαντικά ψηφία του byte δεδομένων, που είναι αποθηκευμένο στη θέση μνήμης $400400, αφήνοντας ανεπηρέαστα τα τέσσερα περισσότερο σημαντικά. Επίσης, θα αντιστρέφει τα τέσσερα λιγότερο σημαντικά ψηφία του καταχωρητή D0 και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400404.***

```
            ORG  $400400
NUM         DC.L $12345678
RESULT      DS.L 1

            ORG      $400410
SUBRTN      MOVE.L NUM,D0
            EORI.B   #$0F,NUM
            EORI.L   #$0000000F,D0
            MOVE.L D0,RESULT
            RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                              ORG       $400400
2  00400400 12345678          NUM:       DC.L      $12345678
3  00400404 00000004          RESULT:    DS.L      1
4
5  00400410                              ORG       $400410
6  00400410 203900400400      SUBRTN:    MOVE.L    NUM,D0
7  00400416 0A39000F00400400             EORI.B    #$0F,NUM
8  0040041E 0A800000000F                 EORI.L    #$0000000F,D0
9  00400424 23C000400404                 MOVE.L    D0,RESULT
10 00400410                              END       $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 12 34 56 78 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 20 39 00 40 04 00 0A 39 00 0F 00 40 04 00 0A 80**

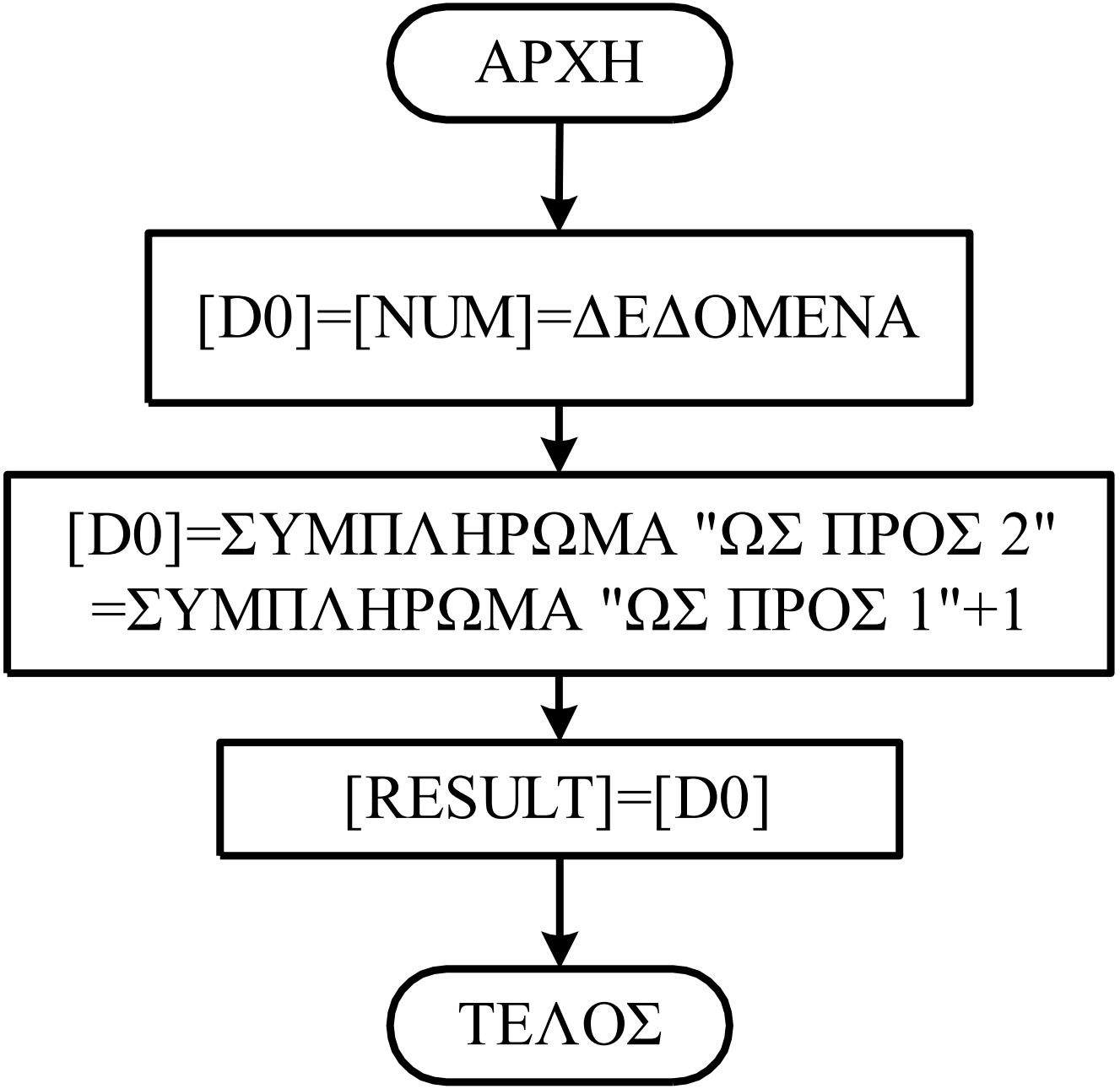
**400420 00 00 00 0F 23 C0 00 40 04 04 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 1D 34 56 78 12 34 56 77 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.17

***Να γραφτεί μια υπορουτίνα που θα δίνει το συμπλήρωμα ¨ως προς δύο¨ της μακριάς λέξης, που βρίσκεται στη θέση μνήμης $400400. Το αποτέλεσμα να αποθηκευτεί στη θέση μνήμης $400404.***


ΑΡΧΗ
[D0]=[NUM]=ΔΕΔΟΜΕΝΑ
[D0]=ΣΥΜΠΛΗΡΩΜΑ "ΩΣ ΠΡΟΣ 2"
=ΣΥΜΠΛΗΡΩΜΑ "ΩΣ ΠΡΟΣ 1"+1
[RESULT]=[D0]
ΤΕΛΟΣ

```
            ORG   $400400
NUM         DC.L  $55555555
RESULT      DS.L  1

            ORG       $400410
SUBRTN      MOVE.L    NUM,D0
            NOT.L     D0
            ADDQ.L    #1,D0
            MOVE.L    D0,RESULT
            RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
1  00400400                              ORG     $400400
2  00400400 55555555          NUM:       DC.L    $55555555
3  00400404 00000004          RESULT:    DS.L    1
4  00400410                              ORG     $400410
5  00400410 203900400400      SUBRTN:    MOVE.L  NUM,D0
6  00400416 4680                         NOT.L   D0
7  00400418 5240                         ADDQ    #1,D0
8  0040041A 23C000400404                 MOVE.L  D0,RESULT
9  00400410                              END     $400410
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 55 55 55 55 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 20 39 00 40 04 00 46 80 52 40 23 C0 00 40 04 04**

**400420 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 55 55 55 55 AA AA AA AB 00 00 00 00 00 00 00 00**

# NOT vs NEG

**Πριν την εκτέλεση της εντολής**

**[D0]=%11100111**

**NEG.B D0** **Εκτελεί αφαίρεση «Συμπληρώματος ως προς 2» του 0-[D0]**

**Μετά την εκτέλεση της εντολής**

**[D0]=%00011001 (XNZVC = 10001)**

**NOT.B D0** **Εκτελεί την λογική πράξη NOT για κάθε ζευγάρι bit**

**Μετά την εκτέλεση της εντολής**

**[D0]=%00011000 (XNZVC = -0000)**

# Εντολές Ολίσθησης και Περιστροφής

# LSL.W (A0)

**LSx Dx,Dy**
**LSx #Imm,Dy**
**LSx <ΕΑ>**

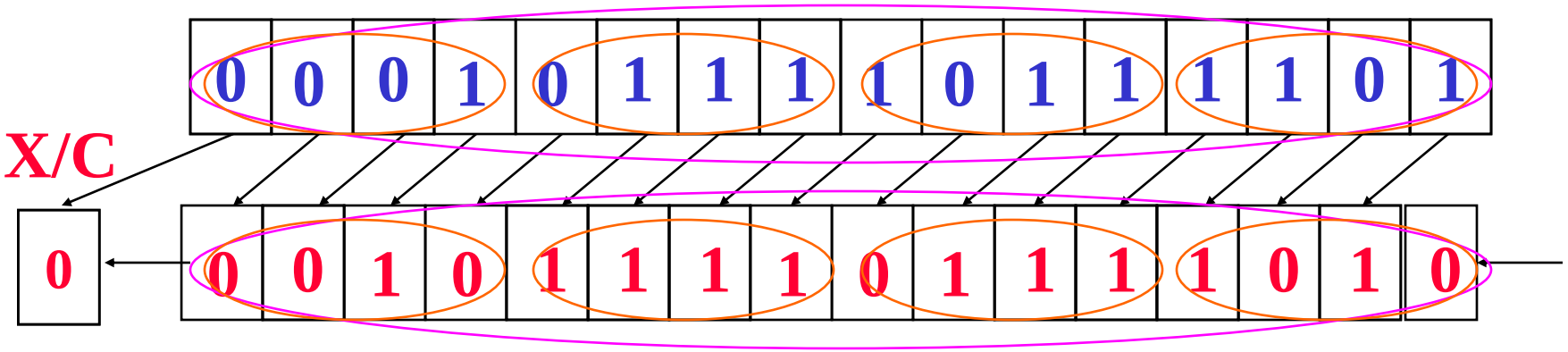
Αν δεν ορίζεται η τιμή της μετατόπισης τότε έχουμε ολίσθηση κατά 1 θέση

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400500, [$400500]=$17BD**

**Μετά την εκτέλεση της εντολής**

**[$00400500]=$2F7A**




V=0

# LSL.W #2,D0

**LSx Dx,Dy**
**LSx #Imm,Dy**
**LSx <ΕΑ>**

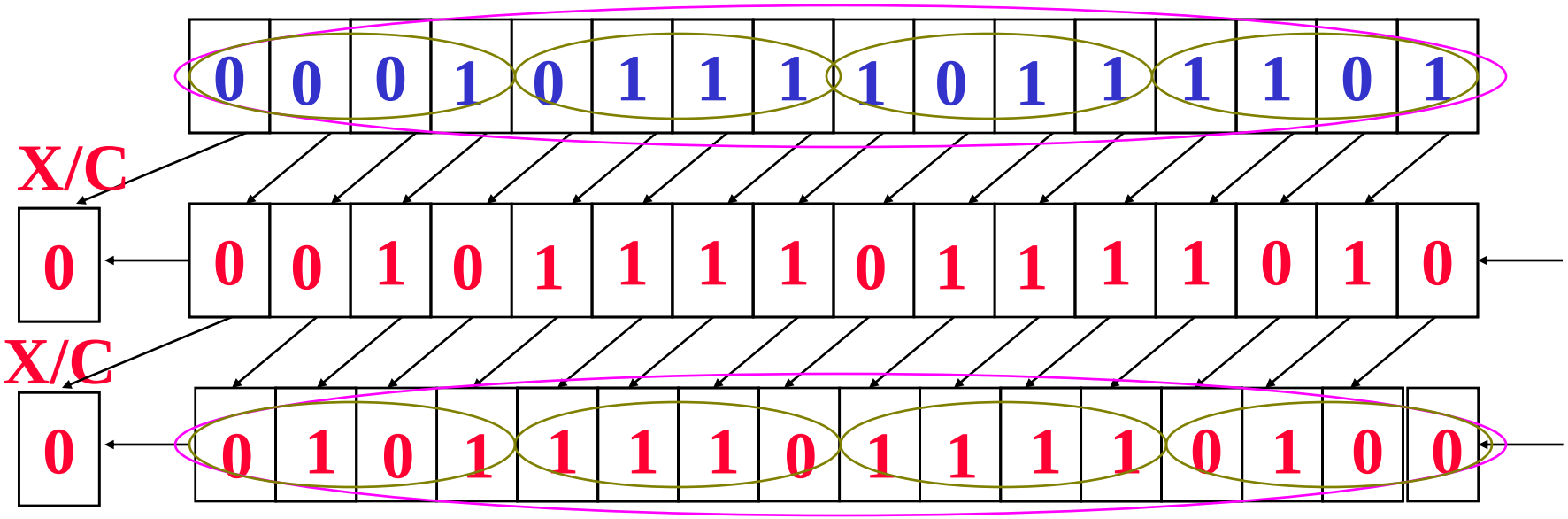
Η τιμή του Immediate μπορεί να είναι από 1 έως 8

## Πριν την εκτέλεση της εντολής

**[D0]=$BBFF17BD**

## Μετά την εκτέλεση της εντολής

**[D0]=$BBFF5EF4**




V=0

# LSL.L D0,D4

**LSx Dx,Dy**
**LSx #Imm,Dy**
**LSx <ΕΑ>**

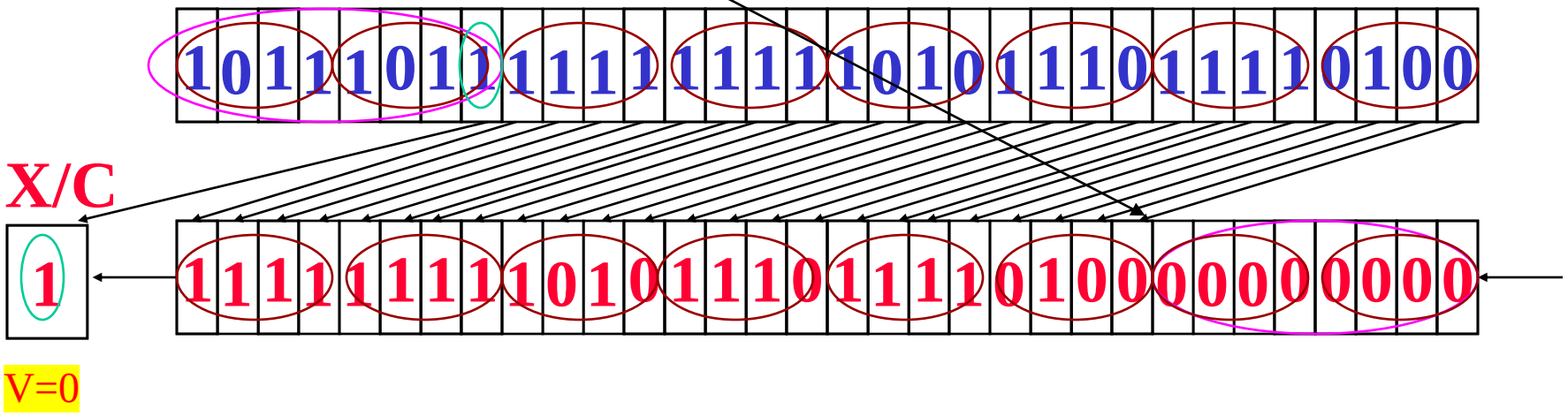
## Πριν την εκτέλεση της εντολής

## [D0]=$0000008, [D4]=$BBFFAEF4

8 Mod64 = 8 θέσεις
άρα από 0 … 63

## Μετά την εκτέλεση της εντολής

## [D4]=$FFAEF400



# ASL, ASR ARITHMETIC SHIFT

**Οι εντολές αυτές λειτουργούν με τον ίδιο τρόπο όπως και οι εντολές LSL και LSR με τη μόνη διαφορά ότι:**

1. **Ενώ στην εντολή LSL ο δείκτης υπερχείλισης V είναι πάντα "0". Στην ASL παίρνει τιμές ανάλογα με το αν έχουμε αλλαγή πρόσημού κατά την ολίσθηση, δηλ. αν αλλάζει το MSB bit.**
2. **Στην εντολή ASR το περισσότερο σημαντικό ψηφίο ανατροφοδοτείται στη θέση του διατηρώντας έτσι το πρόσημο της ψηφιολέξης.**
3. **Η ASL πολλαπλασιάζει με το $2^n$ (n το πλήθος των μετατοπίσεων, το αποτέλεσμα ερμηνεύεται ως ένας αριθμός σε συμπλήρωμα ως προς 2).**
4. **Η ASR διαιρεί με το $2^n$ (n το πλήθος των μετατοπίσεων, το αποτέλεσμα ερμηνεύεται ως ένας αριθμός σε συμπλήρωμα ως προς 2).**
5. **Όταν εφαρμόζονται σε δεδομένα μιας θέσης μνήμης τότε όλες οι εντολές ολίσθησης του M68000 εφαρμόζονται σε word.**

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000002, [D3]=$BBFF97BD**

**Μετά την εκτέλεση της εντολής**

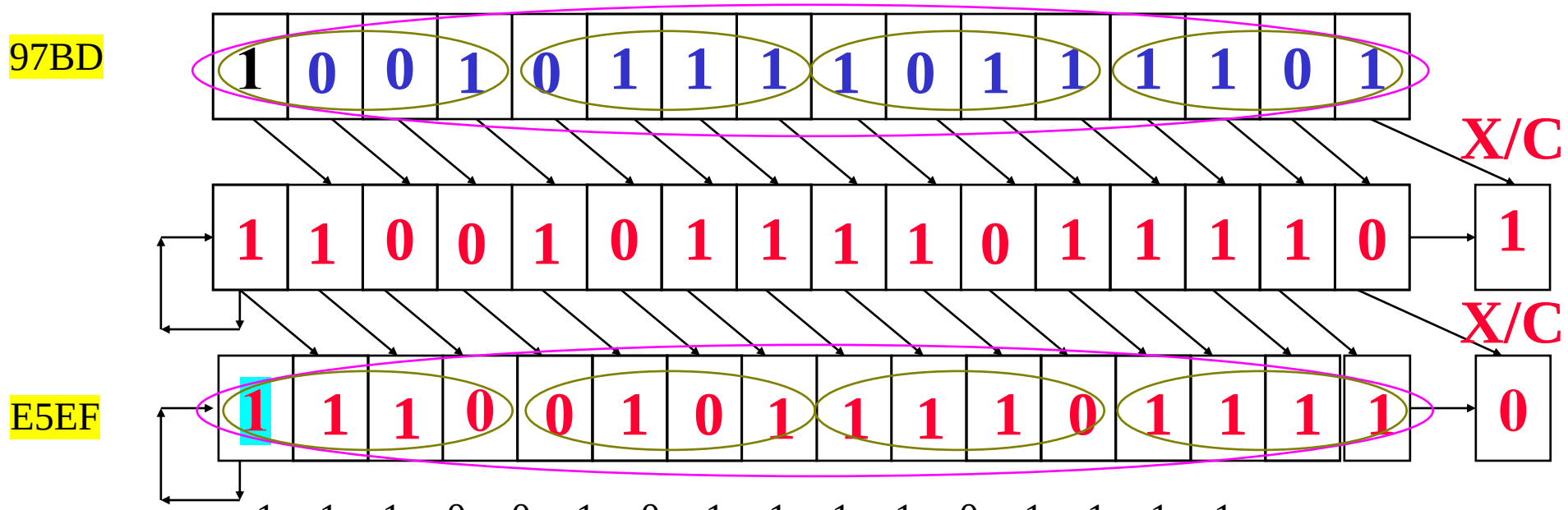
**[D3]=$BBFFE5EF**

**ASx Dx,Dy**
**ASx #Imm,Dy**
**ASx <EΔ>**

- 26.691

- 6.673

# ASR.W D0,D3




1 1 1 0 0 1 0 1 1 1 1 0 1 1 1 1

0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 + 1

\- 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 1 = - 6.673

# ROL.W (A0)

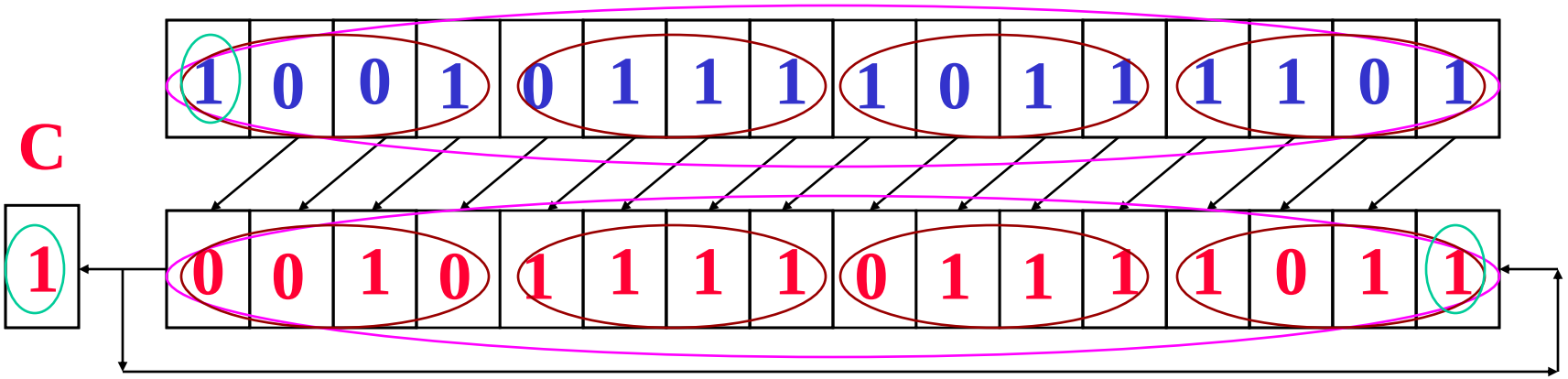
**ROx Dx,Dy**
**ROx #Imm,Dy**
**ROx <ΕΔ>**

**Πριν την εκτέλεση της εντολής**

**[A0]=$00400500, [$400500]=$97BD**

**Μετά την εκτέλεση της εντολής**

**[$400500]=$2F7B**



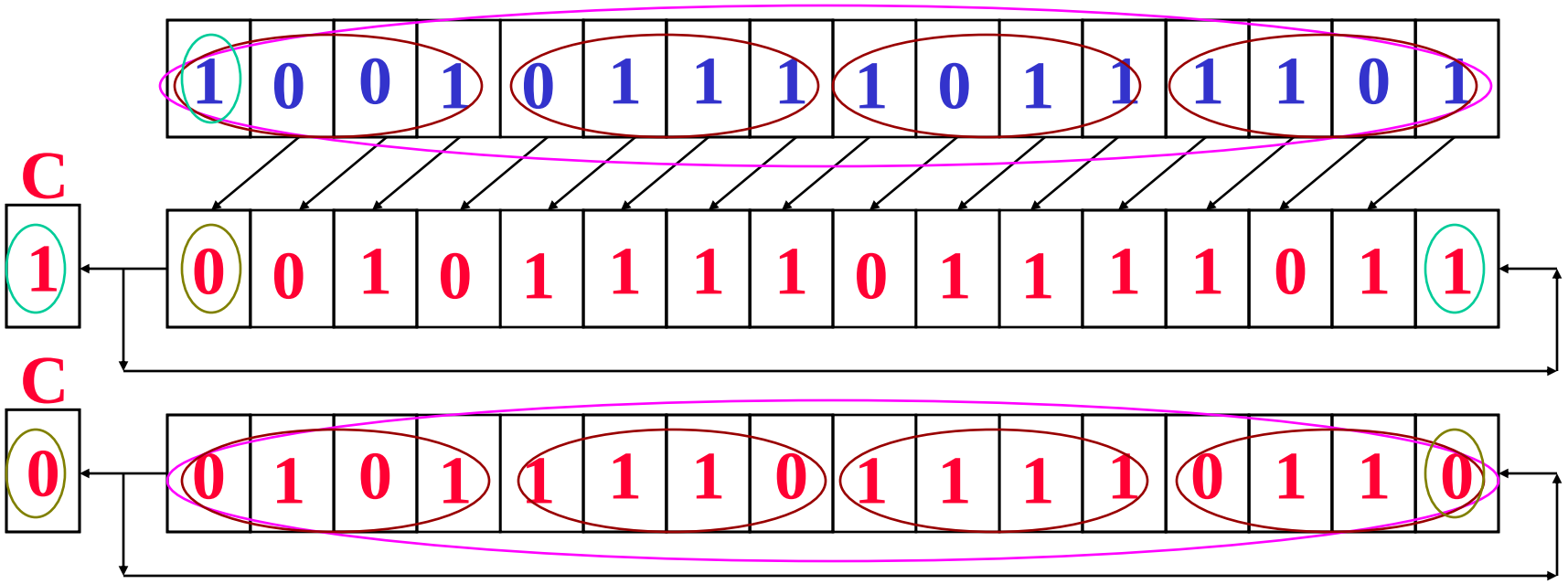
# ROL.W #2,D0

**Πριν την εκτέλεση της εντολής**

**[D0]=$BBFF97BD**

**Μετά την εκτέλεση της εντολής**

**[D0]=$BBFF5EF6**



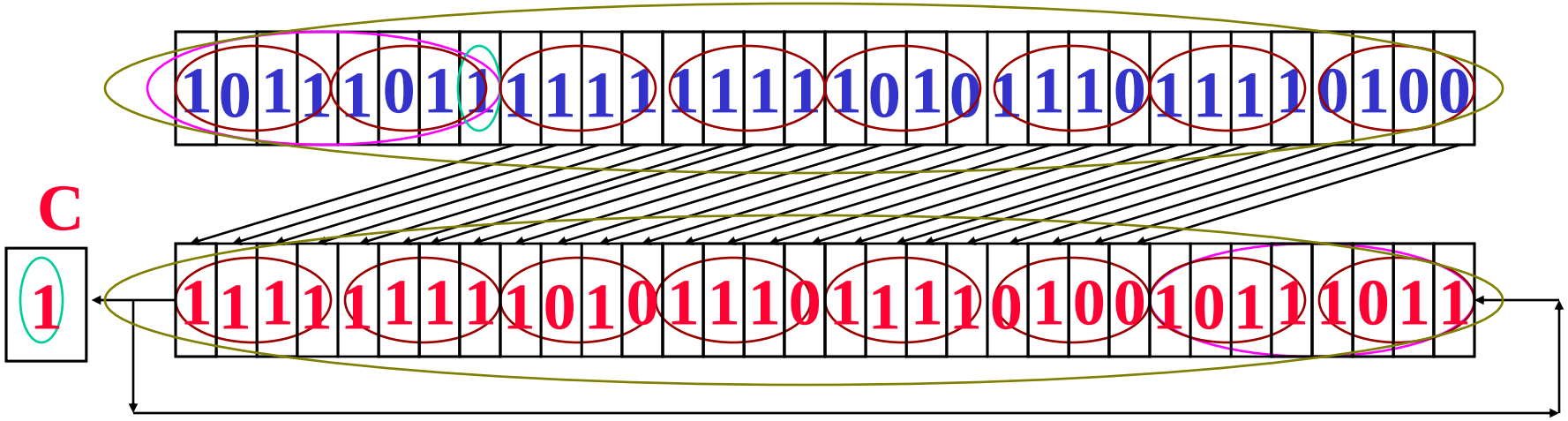
# ROL.L D0,D4

**Πριν την εκτέλεση της εντολής**

**[D0]=$0000008, [D4]=$BBFFAEF4**

**Μετά την εκτέλεση της εντολής**

**[D4]=$FFAEF4BB**



# ROXL.W D0,D4

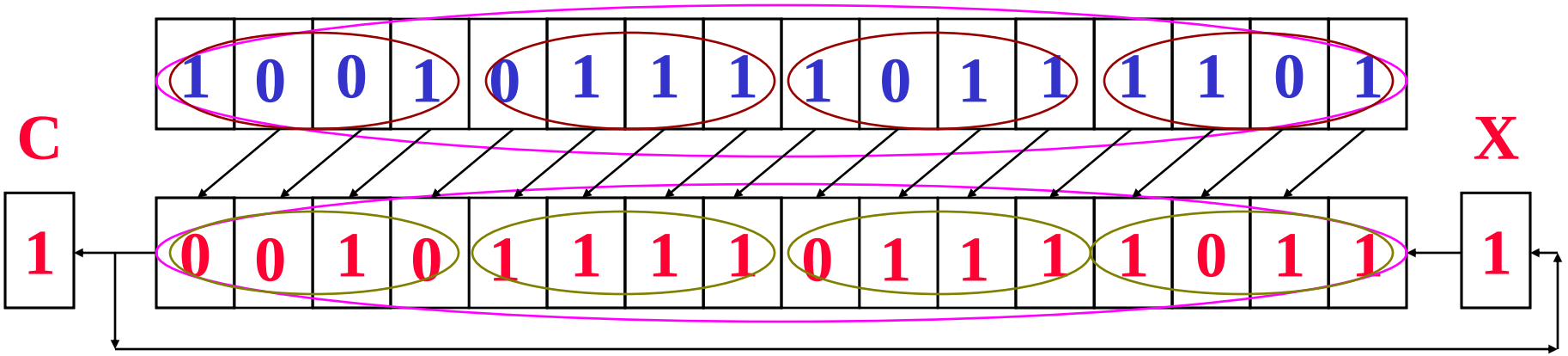
**ROXx Dx,Dy**
**ROXx #Imm,Dy**
**ROXx <ΕΔ>**

**Πριν την εκτέλεση της εντολής**

**[D0]=$00000001, [D4]=$BBFF97BD, X=1**

**Μετά την εκτέλεση της εντολής**

**[D4]=$BBFF2F7B**

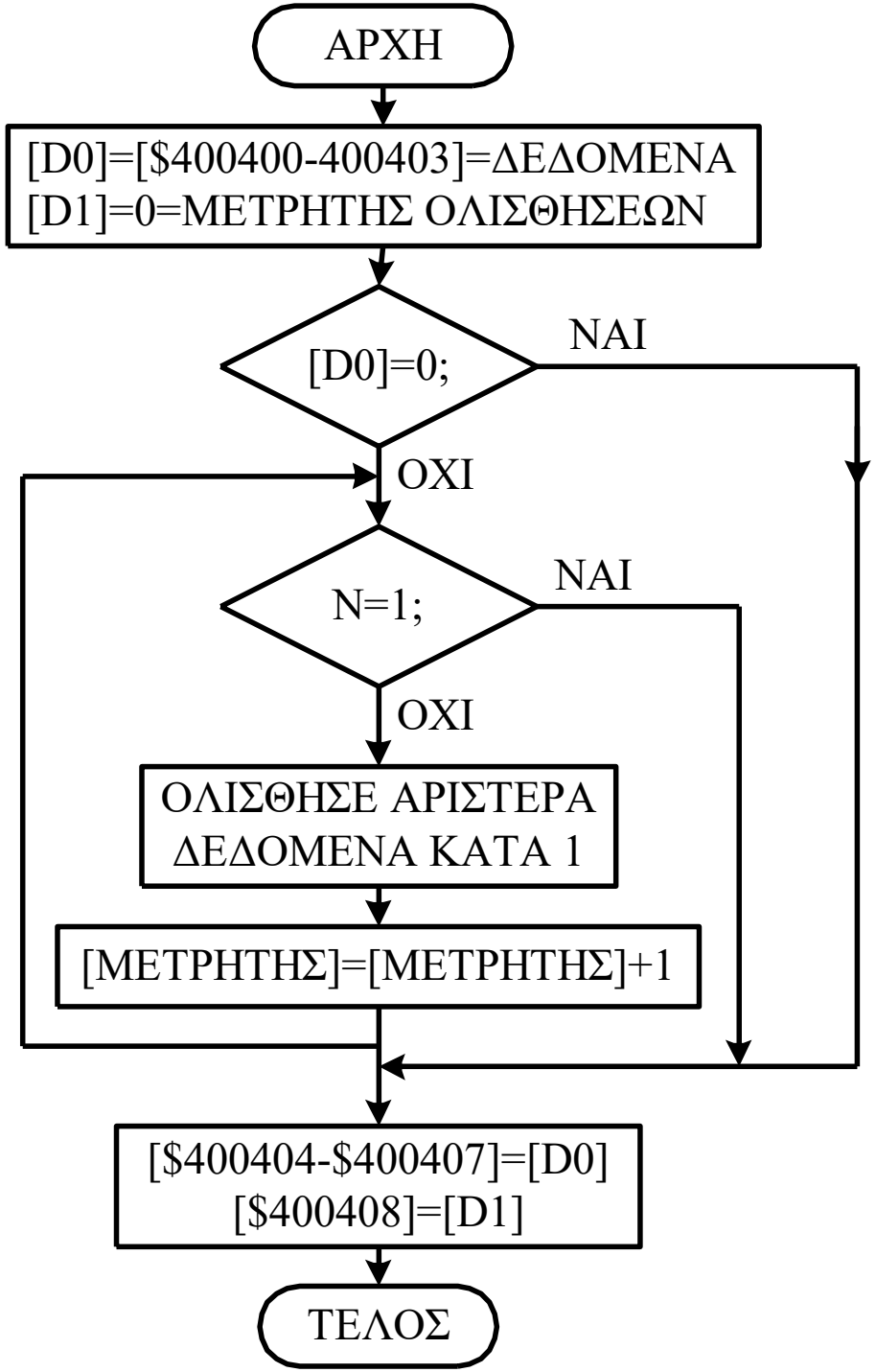


# Παράδειγμα 3.18

***Να γραφτεί μια υπορουτίνα που θα ολισθαίνει προς τα αριστερά τη μακριά λέξη που βρίσκεται στη θέση μνήμης $400400 μέχρις ότου το περισσότερο σημαντικό ψηφίο είναι 1.***

***Το αποτέλεσμα να αποθηκευτεί στη θέση μνήμης $400404 και ο αριθμός των ολισθήσεων που χρειάζονται, να αποθηκευτεί στη θέση μνήμης $400408.***

***Αν τα περιεχόμενα των θέσεων μνήμης $400400-$400403 είναι μηδέν να καθαρίζονται οι θέσεις μνήμης $400404-$400408.***


ΑΡΧΗ
[D0]=[$400400-400403]=ΔΕΔΟΜΕΝΑ
[D1]=0=ΜΕΤΡΗΤΗΣ ΟΛΙΣΘΗΣΕΩΝ
[D0]=0;
ΝΑΙ
ΟΧΙ
N=1;
ΝΑΙ
ΟΧΙ
ΟΛΙΣΘΗΣΕ ΑΡΙΣΤΕΡΑ
ΔΕΔΟΜΕΝΑ ΚΑΤΑ 1
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]+1
[$400404-$400407]=[D0]
[$400408]=[D1]
ΤΕΛΟΣ

| Instruction | Description |
|---|---|
| **BRA** | **BRA (branch always)** implements an unconditional branch, relative to the PC. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, BRA cannot be used. |
| **JMP** | **JMP (jump)** is like BRA. The only difference is that BRA uses only relative addressing, whereas JMP has more addressing modes, including absolute address. |
| **Bcc** | **Bcc (branch on condition code)** is used whenever program execution must follow one of two paths depending on a condition. The condition is specified by the mnemonic **cc**. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, Bcc cannot be used. |
| **JSR**<br>**BSR**<br>**RTS**<br>**RTE**<br>**RTI** | **JSR and BSR** branches to a subroutine. The PC is saved on the stack before loading the PC with the new value.<br>RTS is used to return from the subroutine by restoring the PC from the stack. |

| | cc | Condition | Branch Taken If |
|---|---|---|---|
| | CC | Carry clear | C = 0 |
| | CS | Carry set | C = 1 |
| | NE | Not equal | Z = 0 |
| | EQ | Equal | Z = 1 |
| | PL | Plus | N = 0 |
| | MI | Minus | N = 1 |
| | VC | Overflow clear | V = 0 |
| | VS | Overflow set | V = 1 |
| SIGNED | GE | Greater or equal | NV + N'V' = 0 |
| SIGNED | GT | Greater than | Z'+(NV+N'V')= 1 |
| SIGNED | LE | Less or equal | Z+(N'V+NV') = 1 |
| SIGNED | LT | Less than | N'V + NV' = 1 |
| UNSIGNED | HS | Higher or same | C = 0 |
| UNSIGNED | LO | Lower | C = 1 |
| UNSIGNED | HI | Higher | C'Z' = 1 |
| UNSIGNED | LS | Lower or same | C + Z=1 |

```
          ORG     $400400
NUM       DC.L    $020753EF
RESULT    DS.L    1
SHIFTS    DS.B    1

          ORG     $400410
          CLR.L   D1
          MOVE.L  NUM,D0
          BEQ     SOVER
LOOP      BMI     SOVER
          ADDI.B  #1,D1
          LSL.L   #1,D0
          BRA     LOOP
SOVER     MOVE.L  D0,RESULT
          MOVE.B  D1,SHIFTS
          RTS
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 02 07 53 EF 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 42 81 20 39 00 40 04 00 67 00 00 0C 6B 00 00 08**

**400420 52 41 E3 88 60 F6 23 C0 00 40 04 04 13 C1 00 40**

**400430 04 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

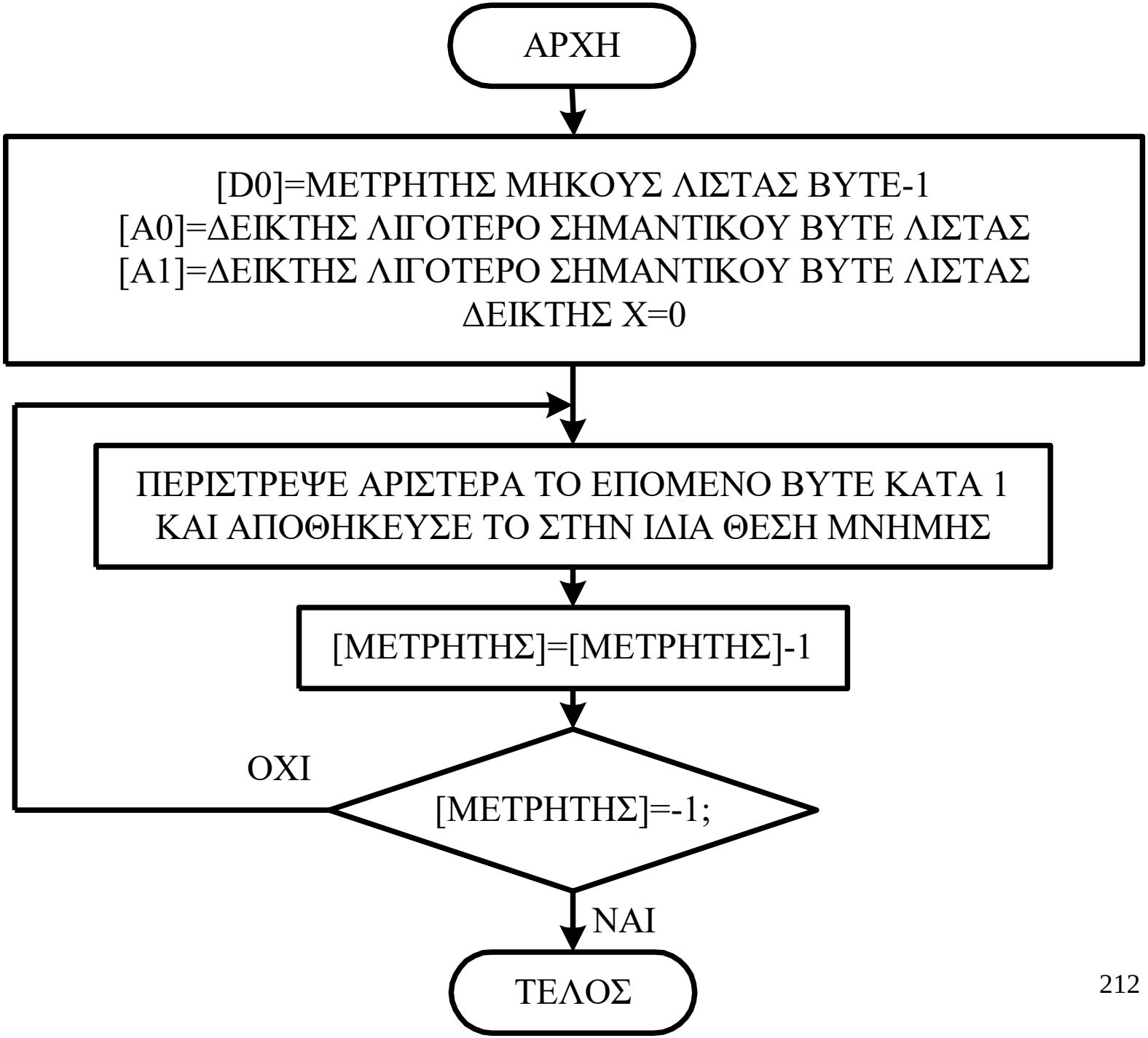
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 02 07 53 EF 81 D4 FB C0 06 00 00 00 00 00 00 00**

# Παράδειγμα 3.19

***Να γραφτεί μια υπορουτίνα που θα περιστρέφει προς τα αριστερά ένα μη προσημασμένο αριθμό πολλών byte που είναι αποθηκευμένος στη μνήμη, με το περισσότερο σημαντικό byte να ξεκινά από τη θέση μνήμης $400401.***

***Το μήκος του αριθμού σε byte είναι αποθηκευμένο στη θέση μνήμης $400400.***


ΑΡΧΗ
[D0]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ ΒΥΤΕ-1
[A0]=ΔΕΙΚΤΗΣ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΛΙΣΤΑΣ
[A1]=ΔΕΙΚΤΗΣ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΛΙΣΤΑΣ
ΔΕΙΚΤΗΣ Χ=0
ΠΕΡΙΣΤΡΕΨΕ ΑΡΙΣΤΕΡΑ ΤΟ ΕΠΟΜΕΝΟ ΒΥΤΕ ΚΑΤΑ 1
ΚΑΙ ΑΠΟΘΗΚΕΥΣΕ ΤΟ ΣΤΗΝ ΙΔΙΑ ΘΕΣΗ ΜΝΗΜΗΣ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]=-1;
ΝΑΙ
ΤΕΛΟΣ

```
                ORG     $400400
SHIFTS          DC.B    4
NUMS            DC.B    $12,$34,$56,$78
                ORG     $400410
SUBRTN          MOVE.B  SHIFTS,D0
                LEA     NUMS,A2
                LEA     (A2,D0),A0
                LEA     (A2,D0),A1
                SUBQ.B  #1,D0
                ANDI.B  #$EF,CCR
LOOP            MOVE.B  -(A0),D1
                ROXL.B  #1,D1
                MOVE.B  D1,-(A1)
                DBRA    D0,LOOP
                END SUBRTN
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 04 12 34 56 78 00 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

**400410 10 39 00 40 04 00 45 F9 00 40 04 01 41 F2 08 00**

**400420 43 F2 08 00 53 00 02 3C 00 EF 12 20 E3 11 13 01**

**400430 51 C8 FF F8 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

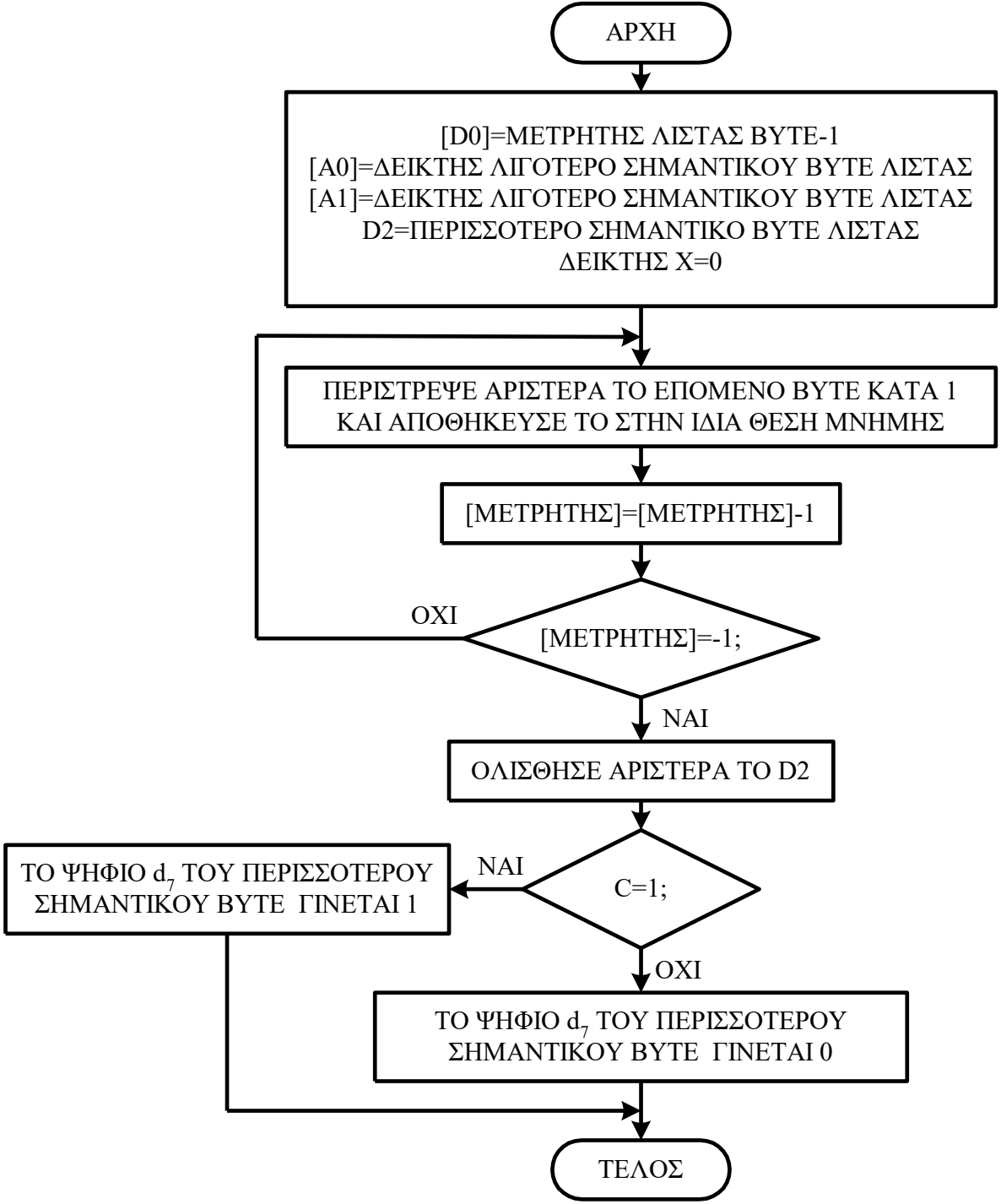
**400400 04 24 68 AC F0 00 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.20

***Να γραφτεί μια υπορουτίνα που θα περιστρέφει προς τα αριστερά έναν προσημασμένο αριθμό πολλών byte που είναι αποθηκευμένος στη μνήμη, με το περισσότερο σημαντικό byte να ξεκινά από την θέση μνήμης $400401.***

***Το μήκος του αριθμού σε byte είναι αποθηκευμένο στη θέση μνήμης $400400.***

***Ο αριθμός των ολισθήσεων είναι ίδιος με το μήκος του αριθμού σε byte.***


ΑΡΧΗ
[D0]=ΜΕΤΡΗΤΗΣ ΛΙΣΤΑΣ ΒΥΤΕ-1
[A0]=ΔΕΙΚΤΗΣ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΛΙΣΤΑΣ
[A1]=ΔΕΙΚΤΗΣ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΛΙΣΤΑΣ
D2=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ ΒΥΤΕ ΛΙΣΤΑΣ
ΔΕΙΚΤΗΣ X=0
ΠΕΡΙΣΤΡΕΨΕ ΑΡΙΣΤΕΡΑ ΤΟ ΕΠΟΜΕΝΟ ΒΥΤΕ ΚΑΤΑ 1
ΚΑΙ ΑΠΟΘΗΚΕΥΣΕ ΤΟ ΣΤΗΝ ΙΔΙΑ ΘΕΣΗ ΜΝΗΜΗΣ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]=-1;
ΝΑΙ
ΟΛΙΣΘΗΣΕ ΑΡΙΣΤΕΡΑ ΤΟ D2
ΤΟ ΨΗΦΙΟ $d_7$ ΤΟΥ ΠΕΡΙΣΣΟΤΕΡΟΥ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΓΙΝΕΤΑΙ 1
ΝΑΙ
C=1;
ΟΧΙ
ΤΟ ΨΗΦΙΟ $d_7$ ΤΟΥ ΠΕΡΙΣΣΟΤΕΡΟΥ ΣΗΜΑΝΤΙΚΟΥ ΒΥΤΕ ΓΙΝΕΤΑΙ 0
ΤΕΛΟΣ

```
                ORG     $400400
SHIFTS          DC.B    4
NUMS            DC.B    $77,$65,$43,$81
                ORG     $400410
SUBRTN          CLR .L  D0
                MOVE.B  SHIFTS,D0
                LEA     NUMS,A2
                MOVE.B  (A2),D2
                LEA     (A2,D0.L),A0
                LEA     (A2,D0.L),A1
                SUBQ.B  #1,D0
                ANDI.B  $EF,CCR
LOOP            MOVE.B  -(A0),D1
                ROXL.B  #1,D1
                MOVE.B  D1,-(A1)
                DBRA    D0,LOOP
SIGN            LSL.B   #1,D2
                BCC     POSIT
                ORI.B   #$80,(A2)
                BRA     FIN
POSIT           ANDI.B  #$7F,(A2)
FIN             RTS
```

α. Περιοχή μνήμης δεδομένων πριν την εκτέλεση του προγράμματος:

**400400 04 77 65 43 81 00 00 00 00 00 00 00 00 00 00**

β. Περιοχή κώδικα

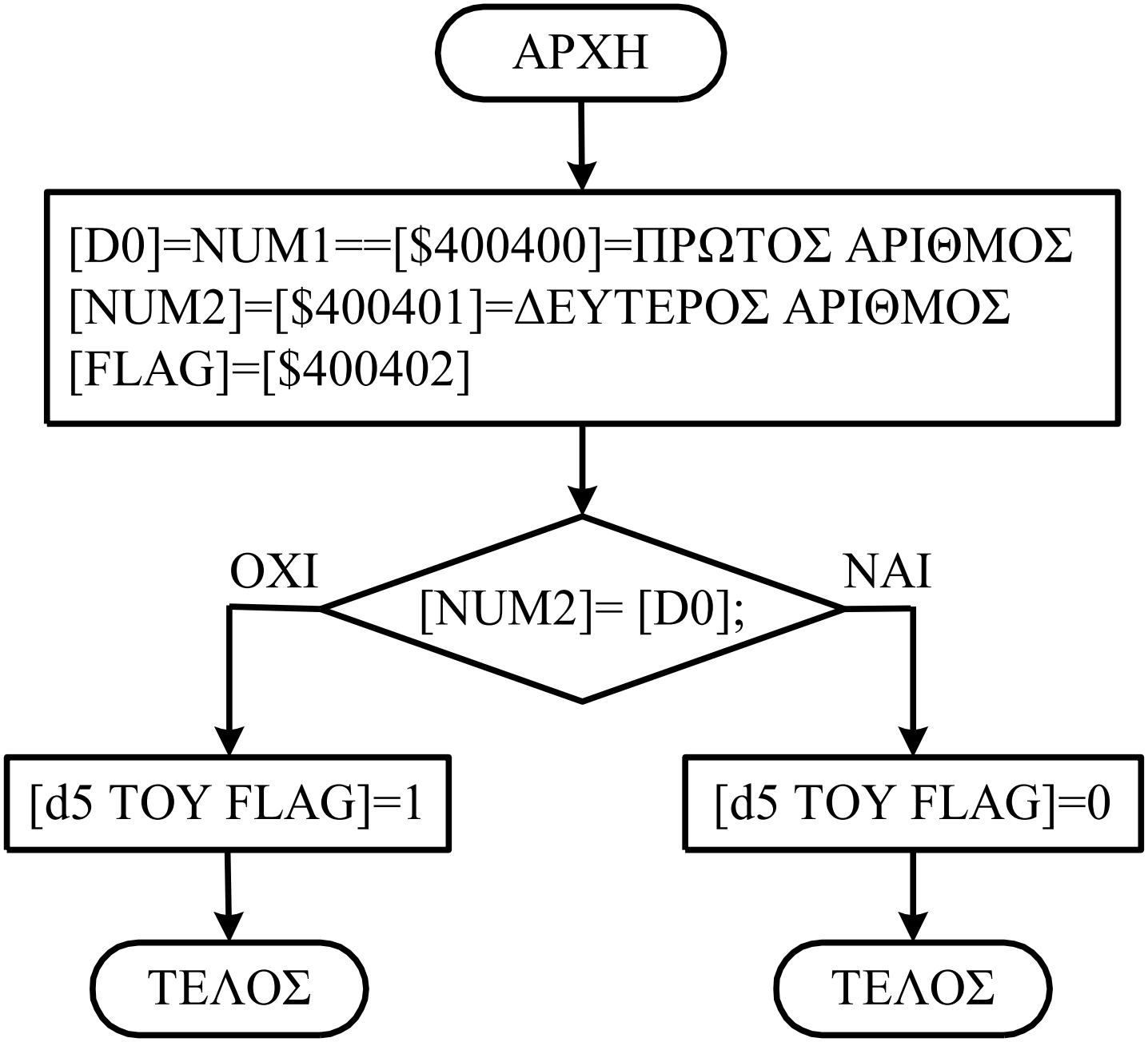
**400410 42 80 10 39 00 40 04 00 45 F9 00 40 04 01 14 12**
**400420 41 F2 08 00 43 F2 08 00 53 00 02 3C 00 EF 12 20**
**400430 E3 11 13 01 51 C8 FF F8 E3 0A 64 00 00 0A 00 12**
**400440 00 80 60 00 00 06 02 12 00 7F 00 00 00 00 00 00**

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

**400400 04 6E CA 87 02 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.22

***Να γραφτεί μια υπορουτίνα που θα κάνει 0 το ψηφίο d5 του byte που είναι αποθηκευμένο στη θέση μνήμης $400402, αν τα byte των θέσεων μνήμης $400400 και $400401 είναι ίσα, διαφορετικά να το κάνει 1.***


ΑΡΧΗ
[D0]=NUM1==[$400400]=ΠΡΩΤΟΣ ΑΡΙΘΜΟΣ
[NUM2]=[$400401]=ΔΕΥΤΕΡΟΣ ΑΡΙΘΜΟΣ
[FLAG]=[$400402]
ΟΧΙ
[NUM2]= [D0];
ΝΑΙ
[d5 ΤΟΥ FLAG]=1
[d5 ΤΟΥ FLAG]=0
ΤΕΛΟΣ
ΤΕΛΟΣ

```
                ORG  $400400
NUM1            DC.B  $4F
NUM2            DC.B  $5A
FLAG            DS.B   1

                ORG       $400410
SUBRTN          MOVE.B NUM1,D0
                CMP.B    NUM2,D0
                BEQ       EQUAL
                ORI.B      #$20,FLAG
                RTS
EQUAL           ANDI.B    #$DF,FLAG
                RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                                ORG       $400400
 2  00400400 4F                 NUM1:       DC.B      $4F
 3  00400401 5A                 NUM2:       DC.B      $5A
 4  00400402 00000001           FLAG:       DS.B      1
 5
 6  00400410                                ORG       $400410
 7  00400410 103900400400       SUBRTN:     MOVE.B    NUM1,D0
 8  00400416 B03900400401                   CMP.B     NUM2,D0
 9  0040041C 6700000E                       BEQ       EQUAL
10  00400420 0039002000400402               ORI.B     #$20,FLAG
11  00400428 6000000A                       BRA       FIN
12  0040042C 023900DF00400402   EQUAL:      ANDI.B    #$DF,FLAG
13  00400410                    FIN:        END       $400410
```

α. Περιεχόμενο μνήμης δεδομένων πριν απ’ την εκτέλεση της υπορουτίνας.

**400400 4F 5A 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιεχόμενο μνήμης κώδικα.

**400410 10 39 00 40 04 00 B0 39 00 40 04 01 67 00 00 0E**

**400420 00 39 00 20 00 40 04 02 60 00 00 0A 02 39 00 DF**

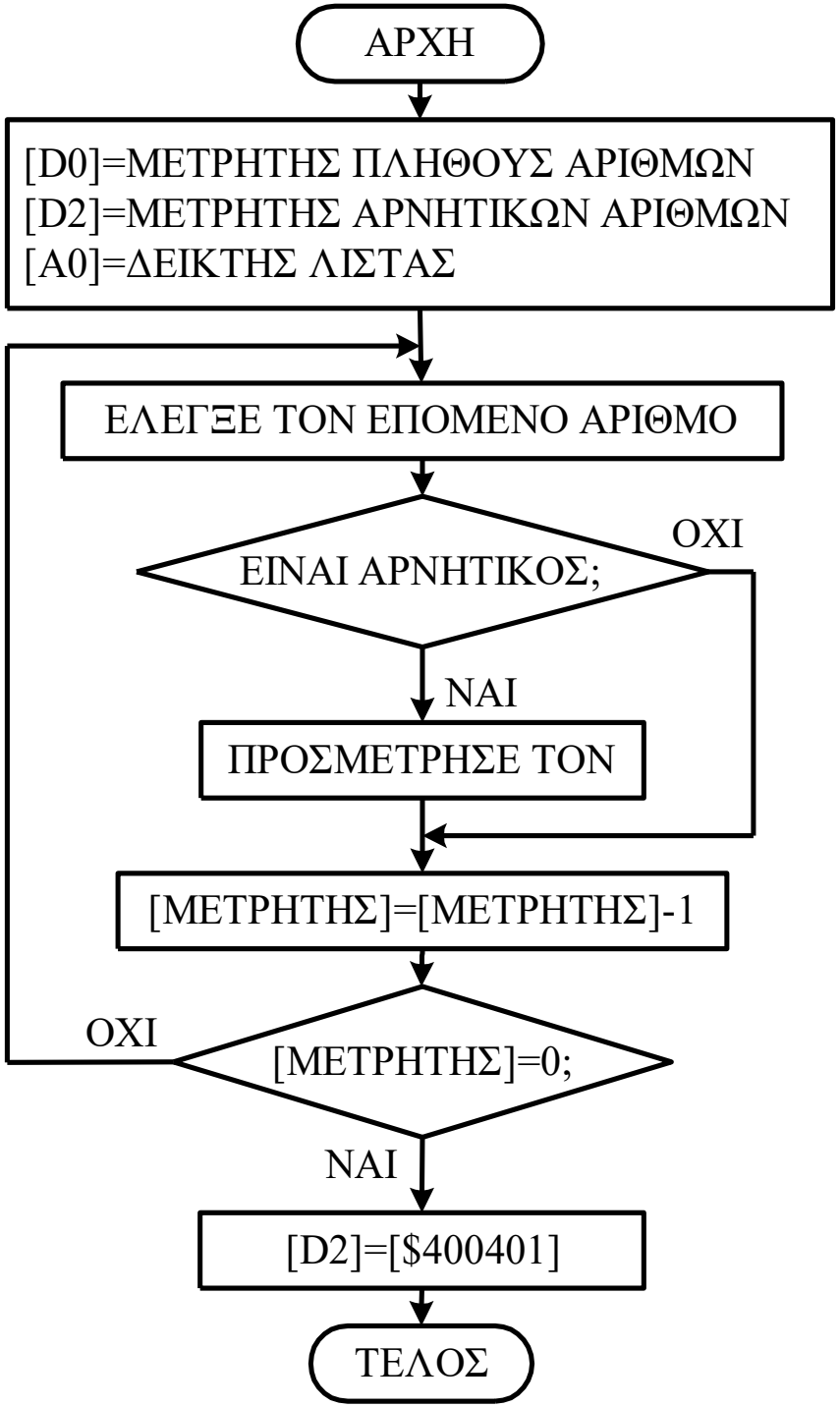
**400430 00 40 04 02 00 00 00 00 00 00 00 00 00 00 00 00**

γ. Περιεχόμενο μνήμης δεδομένων πριν απ’ την εκτέλεση της υπορουτίνας.

**400400 4F 5A 20 00 00 00 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα

***Να γραφτεί μια υπορουτίνα που θα υπολογίζει τους αρνητικούς αριθμούς από ένα πλήθος αριθμών μήκους byte αποθηκευμένων στη μνήμη. Το πλήθος των αριθμών είναι αποθηκευμένο στη θέση μνήμης $400400 και ο πρώτος αριθμός αρχίζει από τη θέση μνήμης $400402. Το αποτέλεσμα να αποθηκευτεί στη θέση μνήμης $400401.***


ΑΡΧΗ
[D0]=ΜΕΤΡΗΤΗΣ ΠΛΗΘΟΥΣ ΑΡΙΘΜΩΝ
[D2]=ΜΕΤΡΗΤΗΣ ΑΡΝΗΤΙΚΩΝ ΑΡΙΘΜΩΝ
[A0]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
ΕΛΕΓΞΕ ΤΟΝ ΕΠΟΜΕΝΟ ΑΡΙΘΜΟ
ΕΙΝΑΙ ΑΡΝΗΤΙΚΟΣ;
ΟΧΙ
ΝΑΙ
ΠΡΟΣΜΕΤΡΗΣΕ ΤΟΝ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
[ΜΕΤΡΗΤΗΣ]=0;
ΟΧΙ
ΝΑΙ
[D2]=[$400401]
ΤΕΛΟΣ

```
                ORG  $400400
COUNT           DC.B  4
RESULT          DS.B   1
NUMS            DC.B  $4F,$F2,$81,$75

                ORG       $400410
SUBRTN          CLR.L     D0
                CLR.L     D2
                MOVE.B    COUNT,D0
                LEA       NUMS,A0
LOOP            MOVE.B    (A0)+,D1
                BPL       POSIT
                ADDQ      #1,D2
POSIT           SUBQ      #1,D0
                BNE       LOOP
                MOVE.B    D2,RESULT
                RTS
```

| | | |
|---|---|---|
| **BPL** | **Plus** | **N = 0** |

| | | |
|---|---|---|
| **BNE** | **Not equal** | **Z = 0** |

# Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:

```
 1  00400400                          ORG       $400400
 2  00400400 04          COUNT:       DC.B      4
 3  00400401 00000001    RESULT:       DS.B     1
 4  00400402 4FF28175    NUMS:        DC.B      $4F,$F2,$81,$75
 5
 6  00400410                          ORG       $400410
 7  00400410 4282        SUBRTN:      CLR.L     D2
 8  00400412 103900400400             MOVE.B    COUNT,D0
 9  00400418 41F900400402             LEA       NUMS,A0
10  0040041E 1218        LOOP:        MOVE.B    (A0)+,D1
11  00400420 6A000004                 BPL       POSIT
12  00400424 5242                     ADDQ      #1,D2
13  00400426 5340        POSIT:       SUBQ      #1,D0
14  00400428 66F4                     BNE       LOOP
15  0040042A 13C200400401             MOVE.B    D2,RESULT
16  00400410                          END       $400410
```

α. Περιεχόμενο μνήμης δεδομένων πριν απ' την εκτέλεση της υπορουτίνας.

**400400 04 00 4F F2 81 75 00 00 00 00 00 00 00 00 00 00**

β. Περιεχόμενο μνήμης κώδικα.

**400410 42 82 10 39 00 40 04 00 41 F9 00 40 04 02 12 18**

**400420 6A 00 00 04 52 42 53 40 66 F4 13 C2 00 40 04 01**

γ. Περιεχόμενο μνήμης δεδομένων πριν απ' την εκτέλεση της υπορουτίνας.

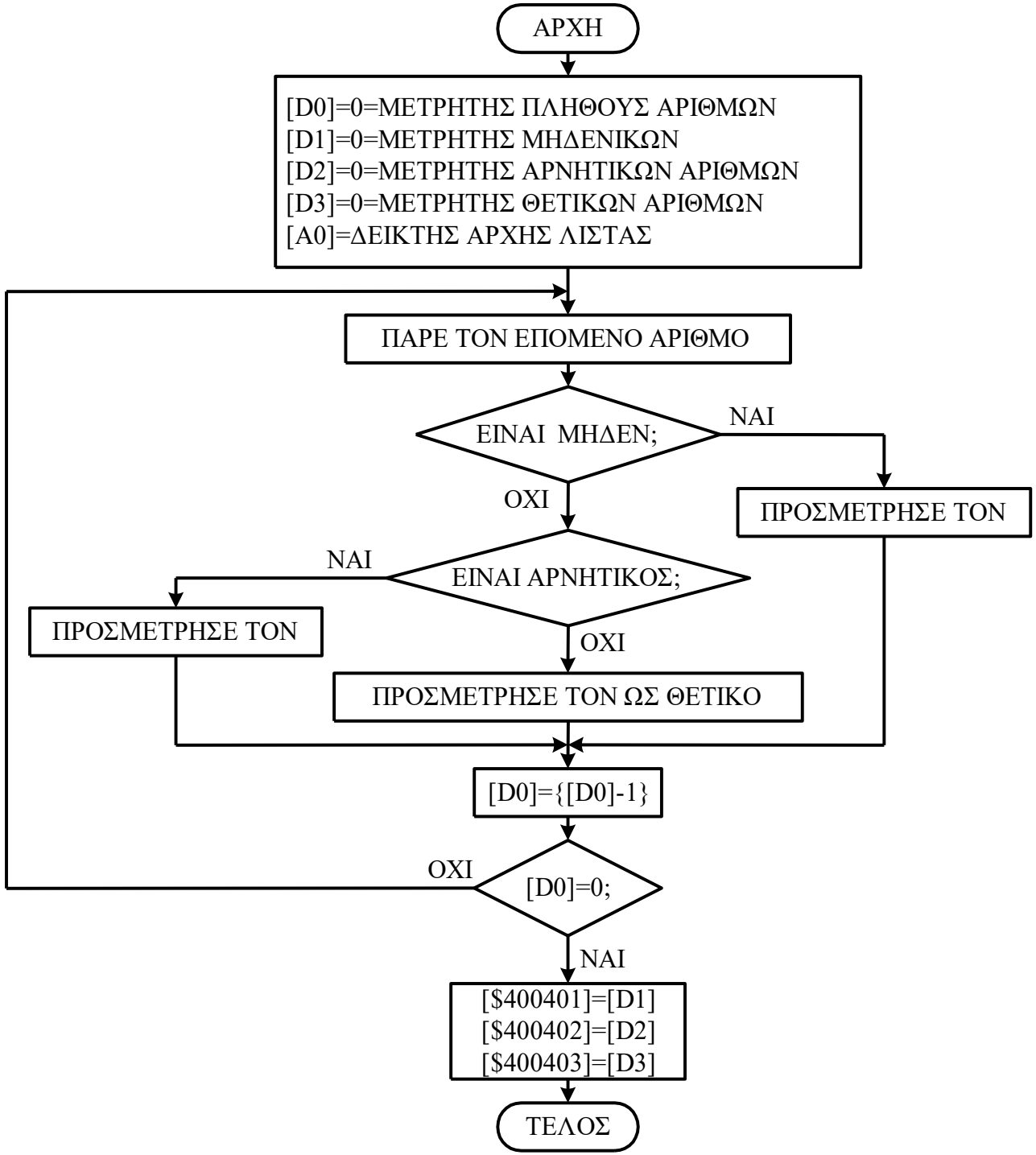
**400400 04 02 4F F2 81 75 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.23

*Να γραφτεί μια υπορουτίνα που θα υπολογίζει τα μηδέν, τους θετικούς και τους αρνητικούς αριθμούς από ένα πλήθος αριθμών αποθηκευμένων στη μνήμη.*

*Το πλήθος των αριθμών είναι αποθηκευμένο στη θέση μνήμης $400400 και ο πρώτος αριθμός αρχίζει από την θέση μνήμης $400404.*

*Τα αποτελέσματα να αποθηκευτούν στις θέσεις μνήμης $400401 (μηδενικά), $400402 (αρνητικοί) και $400403 (θετικοί).*


ΑΡΧΗ
[D0]=0=ΜΕΤΡΗΤΗΣ ΠΛΗΘΟΥΣ ΑΡΙΘΜΩΝ
[D1]=0=ΜΕΤΡΗΤΗΣ ΜΗΔΕΝΙΚΩΝ
[D2]=0=ΜΕΤΡΗΤΗΣ ΑΡΝΗΤΙΚΩΝ ΑΡΙΘΜΩΝ
[D3]=0=ΜΕΤΡΗΤΗΣ ΘΕΤΙΚΩΝ ΑΡΙΘΜΩΝ
[A0]=ΔΕΙΚΤΗΣ ΑΡΧΗΣ ΛΙΣΤΑΣ
ΠΑΡΕ ΤΟΝ ΕΠΟΜΕΝΟ ΑΡΙΘΜΟ
ΕΙΝΑΙ ΜΗΔΕΝ;
ΝΑΙ
ΠΡΟΣΜΕΤΡΗΣΕ ΤΟΝ
ΟΧΙ
ΕΙΝΑΙ ΑΡΝΗΤΙΚΟΣ;
ΝΑΙ
ΠΡΟΣΜΕΤΡΗΣΕ ΤΟΝ
ΟΧΙ
ΠΡΟΣΜΕΤΡΗΣΕ ΤΟΝ ΩΣ ΘΕΤΙΚΟ
[D0]={[D0]-1}
ΟΧΙ
[D0]=0;
ΝΑΙ
[$400401]=[D1]
[$400402]=[D2]
[$400403]=[D3]
ΤΕΛΟΣ

```
        ORG $400400          * Αποθήκευσε τα δεδομένα απ' τη θέση μνήμης $400400 και πάνω.
COUNT   DC.B  6              * Αποθήκευσε τον πλήθος των αριθμών στη θέση COUNT=$400400.
ZEROS   DS.B  1              * Κράτησε ελεύθερη τη θέση μνήμης $400401.=ZERO για μηδενικά.
NEGS    DS.B  1              * Κράτησε ελεύθερη τη θέση μνήμης $400402.=NEG για αρνητικούς.
POSITS  DS.B  1              * Κράτησε ελεύθερη τη θέση μνήμης $400401.=POSIT για θετικούς.
NUMS    DC.B  $4F,$F2,$81,$75,  * Αποθήκευσε απ' τη θέση NUMS και πάνω $4F, $F2, $81, $75,$00,$9A.
        $00,$9A

        ORG $400410          * Αποθήκευσε τον κώδικα απ' τη θέση μνήμης $400410 και πάνω.
SUBRTN  CLR.B D1             * Μηδένισε τον μετρητή των μηδενικών.
        CLR.B D2             * Μηδένισε τον μετρητή των αρνητικών.
        CLR.B D3             * Μηδένισε τον μετρητή των θετικών.
        LEA NUMS,A0          * Κάνε τον A0 δείκτη αρχής της λίστας.
        MOVE.B COUNT,D0      * Πάρε το πλήθος των αριθμών στον D0.
LOOP    MOVE.B (A0)+,D4      * Πάρε τον επόμενο αριθμό στον D4.
        BEQ ZERO             * Αν είναι μηδέν διακλάδωσε στο ZERO.
        BMI NEGAT            * Αν είναι αρνητικός διακλάδωσε στο NEGAT.
        ADDQ #1,D3           * Αν είναι θετικός προσμέτρησέ τον και.
        BRA NEXT             * διακλάδωσε στο NEXT.
ZERO    ADDQ #1,D1           * Είναι μηδέν προσμέτρησέ τον και
        BRA NEXT             * διακλάδωσε στο NEXT.
NEGAT   ADDQ #1,D2           * Είναι αρνητικός προσμέτρησέ τον.
NEXT    SUBQ #1,D0           * Μείωσε το πλήθος των αριθμών κατά 1.
        BNE LOOP             * Αν δεν έχουν ελεγχθεί όλοι οι αριθμοί συνέχισε έλεγχο.
        MOVE.B D1,ZEROS      * Αποθήκευσε τον αριθμό των μηδενικών στη θέση ZERO.
        MOVE.B D2,NEGS       * Αποθήκευσε τον αριθμό των αρνητικών στη θέση NEG.
        MOVE.B D3,POSITS     * Αποθήκευσε τον αριθμό των θετικών στη POSIT.
        RTS                  * Επίστρεψε απ' την υπορουτίνα.
```

```
            ORG $400400
COUNT       DC.B  6
ZEROS       DS.B   1
NEGS        DS.B  1
POSITS      DS.B  1
NUMS        DC.B  $4F,$F2,$81,$75,
            $00,$9A

            ORG $400410
SUBRTN      CLR.B D1
            CLR.B D2
            CLR.B D3
            LEA NUMS,A0
            MOVE.B COUNT,D0
```

```
LOOP    MOVE.B (A0)+,D4
        BEQ ZERO
        BMI NEGAT
        ADDQ #1,D3
        BRA NEXT
ZERO    ADDQ #1,D1
        BRA NEXT
NEGAT   ADDQ #1,D2
NEXT    SUBQ #1,D0
        BNE LOOP
        MOVE.B D1,ZEROS
        MOVE.B D2,NEGS
        MOVE.B D3,POSITS
        RTS
```

α. Περιεχόμενο μνήμης δεδομένων πριν απ’ την εκτέλεση της υπορουτίνας.

**400400 06 00 00 00 4F F2 81 75 00 9A 00 00 00 00 00 00**

β. Περιεχόμενο μνήμης κώδικα.

**400410 42 01 42 02 42 03 41 F9 00 40 04 04 10 39 00 40**
**400420 04 00 18 18 67 00 00 0C 6B 00 00 0E 52 43 60 00**
**400430 00 0A 52 41 60 00 00 04 52 42 53 40 66 E4 13 C1**
**400440 00 40 04 01 13 C2 00 40 04 02 13 C3 00 40 04 03**

γ. Περιεχόμενο μνήμης δεδομένων μετά την εκτέλεση της υπορουτίνας.

**400400 06 01 03 02 4F F2 81 75 00 9A 00 00 00 00 00 00**

# Εντολές Χειρισμού Ψηφίου

# BTST #7,D1

**Πριν την εκτέλεση της εντολής**

**[D1]=$26ECA86=%0010 0110 1110 1100 1010 1000 0110**

**Μετά την εκτέλεση της εντολής**

**Z=0 γιατί το ψηφίο $d_7$ του D1 είναι ¨1¨.**

# BTST D0,D1

**Πριν την εκτέλεση της εντολής**

**[D0]=$0000000A=%0000 0000 0000 0000 0000 0000 0000 1010**

**[D1]=$526ECA86=%0101 0010 0110 1110 1100 1010 1000 0110**

**Μετά την εκτέλεση της εντολής**

**Z=1 γιατί το ψηφίο $d_{10}$ του D1 είναι ¨0¨.**

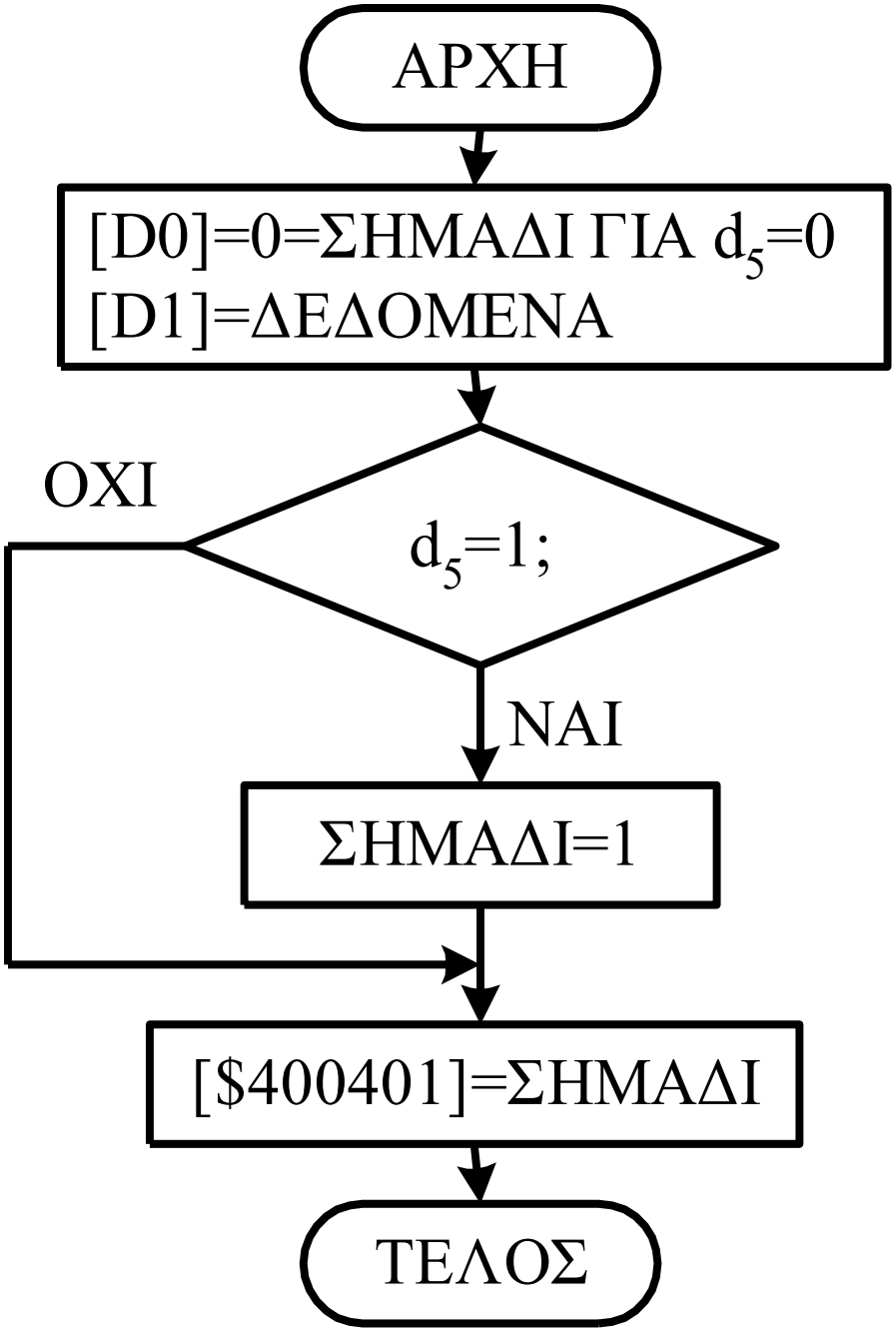
# BSET, BCLR, BCHG,TAS

**Οι εντολές αυτές ελέγχουν το ψηφίο του τελεστέου προορισμού και αντιγράφουν το συμπλήρωμά του στο δείκτη Z, ενώ ταυτόχρονα κάνουν το υπό έλεγχο ψηφίο της ψηφιολέξης ¨1¨ (BSET) ή ¨0¨ (BCLR) ή το αντικαθιστούν με το συμπλήρωμά του (BCHG).**

**Η εντολή TAS διαφέρει από τη BTST στο ότι ελέγχει μια ψηφιολέξη μήκους byte συγκρίνοντας τον τελεστέο με το μηδέν και κάνει ¨0¨ ή ¨1¨ τους δείκτες N και Z ανάλογα με το αποτέλεσμα. Έτσι, ο Δείκτης Z παίρνει την τιμή ¨0¨ αν ο τελεστέος είναι ¨0¨ και ο δείκτης N παίρνει την τιμή του περισσότερου σημαντικού ψηφίου του τελεστέου. Η εντολή αυτή χρησιμοποιείται όταν πρόκειται ο επεξεργαστής να λειτουργήσει σε περιβάλλον πολυεπεξεργασίας.**

# Παράδειγμα 3.25

*Να γραφτεί μια υπορουτίνα που θα κάνει ένα την θέση μνήμης $400401, αν το ψηφίο d5 της θέσης μνήμης $400400 είναι 1, διαφορετικά να την κάνει 0.*


ΑΡΧΗ
[D0]=0=ΣΗΜΑΔΙ ΓΙΑ $d_5$=0
[D1]=ΔΕΔΟΜΕΝΑ
ΟΧΙ
$d_5$=1;
ΝΑΙ
ΣΗΜΑΔΙ=1
[$400401]=ΣΗΜΑΔΙ
ΤΕΛΟΣ

```
        ORG  $400400
NUM     DC.B  $2E
FLAG    DS.B   1

        ORG      $400410
SUBRTN  CLR.L    D0
        MOVE.B NUM,D1
        BTST     #5,D1
        BEQ      DONE
        ORI.B    #1,D0
DONE    MOVE.B D0,FLAG
        RTS
```

**Μετά τη συμβολομετάφραση ο συμβολομεταφραστής θα δώσει την παρακάτω λίστα:**

```
 1  00400400                          ORG       $400400
 2  00400400 2E              NUM:     DC.B      $2E
 3  00400401 00000001        FLAG:    DS.B      1
 4
 5  00400410                          ORG       $400410
 6  00400410 4280            SUBRTN:  CLR.L     D0
 7  00400412 123900400400             MOVE.B    NUM,D1
 8  00400418 08010005                 BTST      #5,D1
 9  0040041C 67000006                 BEQ       DONE
10  00400420 00000001                 ORI.B     #$01,D0
11  00400424 13C000400401    DONE:    MOVE.B    D0,FLAG
12 00400410                          END       $400410
```

α. Περιεχόμενο μνήμης δεδομένων πριν απ' την εκτέλεση της υπορουτίνας.

**400400 2E 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

β. Περιεχόμενο μνήμης κώδικα.

**400410 42 80 12 39 00 40 04 00 08 01 00 05 67 00 00 06**

**400420 00 00 00 01 13 C0 00 40 04 01 00 00 00 00 00 00**

γ. Περιεχόμενο μνήμης δεδομένων πριν απ' την εκτέλεση της υπορουτίνας.

**400400 2E 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

# Παράδειγμα 3.5

***Με βάση το παράρτημα Ι να γίνει η συμβολομετάφραση του παρακάτω προγράμματος.***

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**1. Όταν εκτελείται η εντολή ORG $400400 ο συμβολομεταφραστής τοποθετεί τον εσωτερικό του μετρητή προγράμματος στο $400400.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**2. Όταν εκτελείται η εντολή CLR.W D4 ο συμβολομεταφραστής φορτώνει τον κωδικό της εντολής CLR, που είναι 01000010*ssmmmrrr*,**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Όπου, *ss* είναι τα ψηφία του μεγέθους της λέξης, που η εντολή πρόκειται να επεξεργαστεί (στη προκειμένη περίπτωση θα είναι $01_2$ για μήκος λέξης),**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

***mmm*** **είναι τα ψηφία της μεθόδου διευθυνσιοδότησης που πρόκειται να χρησιμοποιηθεί (στην προκειμένη περίπτωση $000_2$ για άμεση διευθυνσιοδότηση καταχωρητή δεδομένων),**

| Μέθοδος Διευθυνσιοδότησης | Μορφή | Κατάχωρητής |
|---|---|---|
| Dn | 000 | Αριθμός Καταχωρητή |
| An | - | - |
| (An) | 010 | Αριθμός Καταχωρητή |
| (An)+ | 011 | Αριθμός Καταχωρητή |
| -(An) | 100 | Αριθμός Καταχωρητή |
| d(An) | 101 | Αριθμός Καταχωρητή |
| d(An,Xi) | 110 | Αριθμός Καταχωρητή |
| Abs.W | 111 | 000 |
| Abs.L | 111 | 001 |

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

Αφού το σύνολο της εντολής κωδικοποιείται σε μια λέξη ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 2 στην τιμή $400402.

και *rrr* είναι τα ψηφία που δηλώνουν το καταχωρητή δεδομένων προορισμού (στην προκειμένη περίπτωση $100_2$ για τον καταχωρητή δεδομένων D4). Έτσι, στην προκειμένη περίπτωση ο κωδικός εντολής θα είναι:

**CLR.W D4=$0100001001000100_2 = 4244_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**3. Όταν εκτελείται η εντολή BSR $400460 ο συμβολομεταφραστής αφαιρεί την επόμενη φυσική τιμή του μετρητή προγράμματος ($400402+2 ή $400404) από τη διεύθυνση στόχου, που καθορίζεται από την εντολή, προκειμένου να υπολογίσει τη μετατόπιση.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Έτσι, στη προκειμένη περίπτωση, η μετατόπιση θα είναι $400460-$400404=$5C που είναι μια μετατόπιση 8 ψηφίων.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

Αφού το σύνολο της εντολής κωδικοποιείται σε μια λέξη ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 2 στην τιμή $400404.

**Ο κωδικός εντολής της εντολής BSR είναι 01100001*dddddddd*, όπου *dddddddd* είναι τα οκτώ ψηφία του byte μετατόπισης. Άρα στη προκειμένη περίπτωση ο κωδικός της εντολής θα είναι:**

**BSR $400460=$0110000101011100_2=615C_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Η εντολή αυτή χρειάζεται δύο θέσεις μνήμης για να αποθηκευτεί και επομένως ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 4 κάνοντάς το $400404+4=$400408.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Η ετικέτα LOOP θα πάρει την τιμή που έχει εκείνη τη στιγμή ο μετρητής προγράμματος και αφού ο μετρητής προγράμματος έχει την τιμή $400404 αυτή θα είναι και η τιμή της ετικέτας LOOP.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Όταν εκτελείται η εντολή BSR $403000 το περιεχόμενο του μετρητή προγράμματος ($400404+4=$400408) αφαιρείται από τη διεύθυνση στόχου $403000 δίνοντας μία μετατόπιση μήκους λέξης ($403000-$400408=$2BF8).**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

Άρα στη προκειμένη περίπτωση ο κωδικός της εντολής θα είναι:

**BSR $403000=0110000100000000$_2$=6100 2BF8$_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Η εντολή αυτή θα χρειαστεί δύο θέσεις μνήμης για να αποθηκευτεί επομένως ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 4 κάνοντας το $400408+4=$40040C.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**5. Όταν εκτελείται η εντολή ADDI.W #1000, D4 ο συμβολομεταφραστής φορτώνει τον κωδικό της εντολής ADDI, που είναι 00000110*ssmmmrrr*, όπου *ss* θα είναι $01_2$, *mmm* $000_2$ και *rrr* $100_2$.**

| Μέθοδος Διευθυνσιοδότησης | Μορφή | Καταχωρητής |
|---|---|---|
| Dn | 000 | Αριθμός Καταχωρητή |
| An* | 001 | Αριθμός Καταχωρητή |
| (An) | 010 | Αριθμός Καταχωρητή |
| (An)+ | 011 | Αριθμός Καταχωρητή |
| -(An) | 100 | Αριθμός Καταχωρητή |
| d(An) | 101 | Αριθμός Καταχωρητή |
| d(An,Xi) | 110 | Αριθμός Καταχωρητή |
| Abs.W | 111 | 000 |
| Abs.L | 111 | 001 |
| d(PC) | 111 | 010 |
| d(PC,Xi) | 111 | 011 |
| Imm | 111 | 100 |

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Έτσι, στην προκειμένη περίπτωση ο κωδικός εντολής θα είναι:**

**ADDI.W #1000,D4=$0000011001000100_2 = 0644_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

Αφού το σύνολο της εντολής κωδικοποιείται σε δύο λέξεις ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 4 στην τιμή $40040C.

Τα απευθείας δεδομένα $\#1000_{10}$=$03E8 θα τοποθετήθούν απ' το συμβολομεταφραστή αμέσως μετά τον κωδικό της εντολής, σχηματίζοντας την εντολή δύο λέξεων

ADDI.W #1000,D4= $0644 03E8.

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**6. Όταν εκτελείται η εντολή MULU D2,D4, ο συμβολομεταφραστής φορτώνει τον κωδικό της εντολής MULU, που είναι 1100*nnn*011*mmmrrr*, όπου *nnn* θα είναι ο αριθμός του καταχωρητή προορισμού $100_2$, *mmm* $000_2$ και *rrr* $010_2$.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Παρατηρείται ότι δεν χρησιμοποιούνται ψηφία μεγέθους αφού το μέγεθος της εντολής MULU είναι εξ ορισμού .W.**

**Έτσι, στην προκειμένη περίπτωση ο κωδικός εντολής θα είναι:**

**MULU D2,D4=$1100100011000010_2 = C8C2_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

**Ο μετρητής προγράμματος θα αυξήσει το περιεχόμενό του κατά 2 κάνοντας το $40040C+2=$40040E.**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

CMP.B $20(A6),D2, ο συμβολομεταφραστής φορτώνει τον κωδικό της εντολής CMP, που είναι **1011*nnnooommmrrr***, όπου τα νέα ψηφία ***ooo*** αναφέρονται στη μορφή του κώδικα της εντολής. Αυτά τα ψηφία για λειτουργία .B είναι $\boldsymbol{000_2}$. Τα υπόλοιπα ψηφία θα είναι $\boldsymbol{nnn=010_2}$, $\boldsymbol{mmm=101_2}$ και $\boldsymbol{rrr=110_2}$.

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

## Κωδικός εντολής

**CMP.B $20(A6),D2=$1011010000101110_2 = B42E_{16}$**

```
        ORG     $400400
        CLR.W   D4
        BSR     $400460
LOOP    BSR     $403000
        ADDI.W  #1000,D4
        MULU    D2,D4
        CMP.B   $20(A6),D2
        BNE     LOOP
        END
```

Όταν εκτελείται η εντολή **BNE LOOP**, το επόμενο πιθανό περιεχόμενο του μετρητή προγράμματος, που είναι το $400414, αφαιρείται απ' τη διεύθυνση στόχου, που είναι η LOOP και ισούται με $400404, δίνοντας ως αποτέλεσμα $FFF0.

Απ' το αποτέλεσμα της αφαίρεσης φαίνεται ότι η μετατόπιση είναι ένας αρνητικός αριθμός που θα προκαλέσει μία διακλάδωση σε προηγούμενη εντολή. Στη συγκεκριμένη περίπτωση για το συμβολομεταφραστή η μετατόπιση $F0 είναι αρκετή και δε χρειάζονται και τα 16 ψηφία.

Θα χρησιμοποιηθεί, επομένως, η μορφή της εντολής με προσημασμένη μετατόπιση 8 ψηφίων. Ο κωδικός της εντολής διακλάδωσης υπό συνθήκη είναι **0110*ccccddddddddd***, όπου τα ψηφία ***cccc*** παριστάνουν τα τέσσερα ψηφία του κώδικα συνθήκης του καταχωρητή κατάστασης, τα οποία παίρνουν την τιμή $\boldsymbol{0110_2}$ για τη συνθήκη NE, όπως φαίνεται στον αντίστοιχο πίνακα της εντολής Bcc στο παράρτημα Ι. Τα υπόλοιπα 8 ψηφία θα είναι το byte μετατόπισης $F0.

## Κωδικός εντολής BNE LOOP=$0110011011110000_2 = 66F0_{16}$

Η τελευταία εντολή END τερματίζει τη συμβολομετάφραση.