v.11-04-2024

ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

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

Διάλεξη No4:
M68000: Assembly Programs

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:

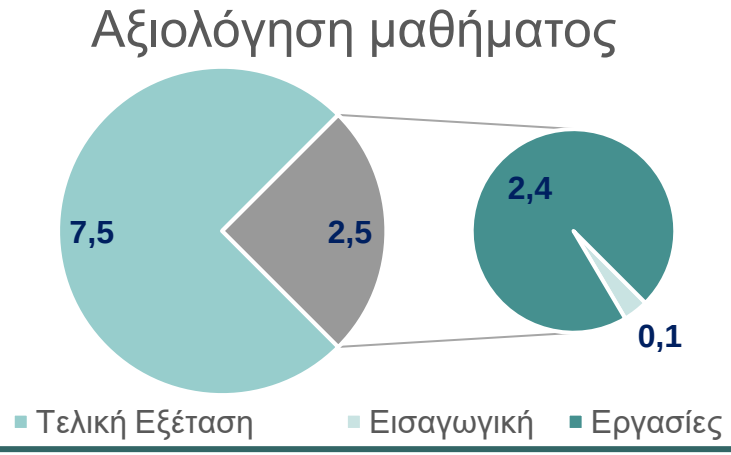
μετά από συνεννόηση με email στο ΦΕ 315 (πάνω από αιθ. Α1)

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/

Κεφάλαιο ΙΙΙ
Προγράμματα -Υπορουτίνες

Υπορουτίνες μετατροπής αριθμών

ΠΑΡΑΔΕΙΓΜΑ 3.26

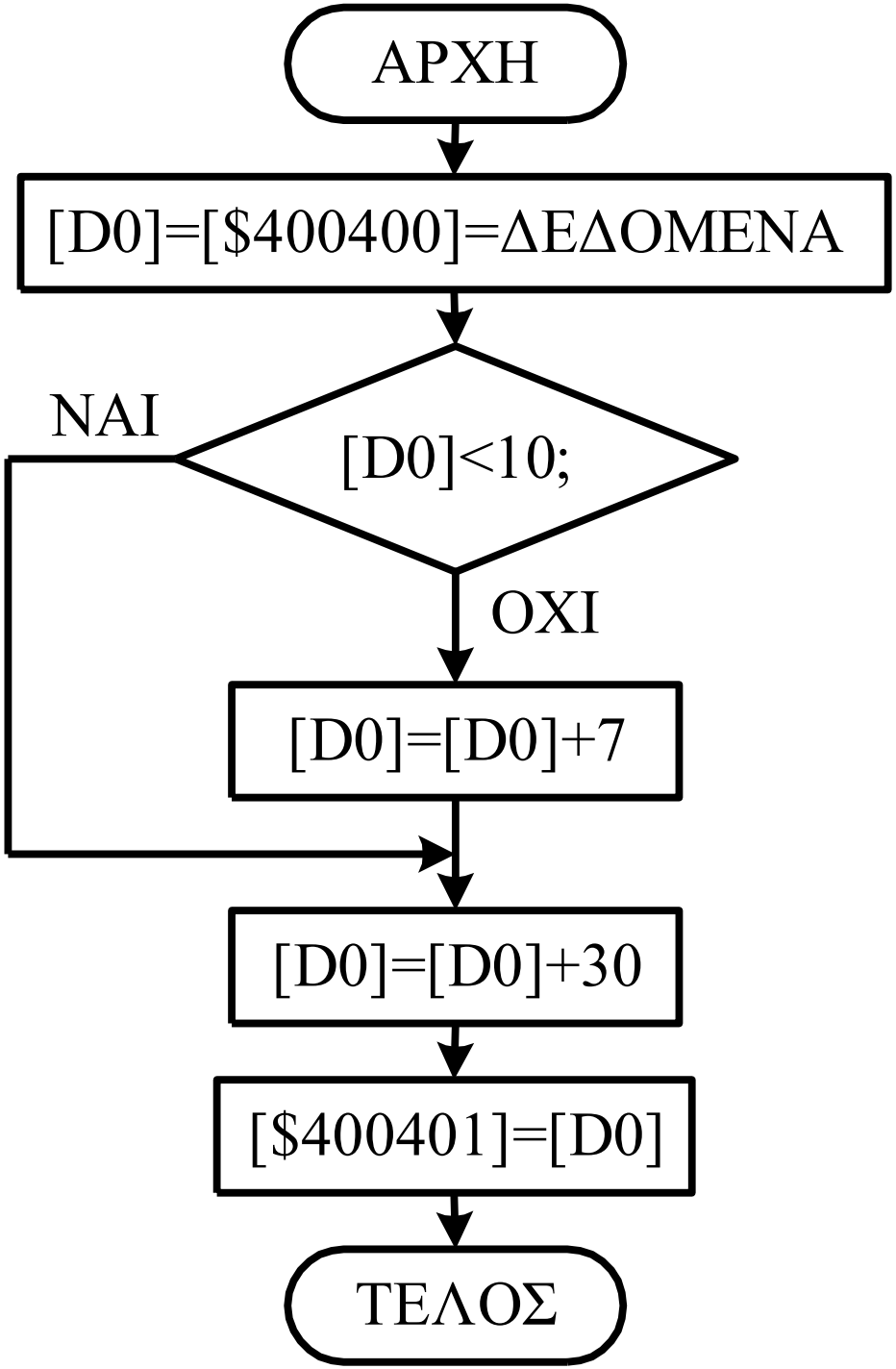
Να γραφτεί μια υπορουτίνα που θα μετατρέπει τα περιεχόμενα της θέσης μνήμης $400400 σε χαρακτήρες ASCII.

Η θέση μνήμης $400400 περιέχει ένα απλό δεκαεξαδικό αριθμό (το περισσότερο σημαντικό nibble είναι μηδέν).

Ο χαρακτήρας ASCII να αποθηκευτεί στη θέση μνήμης $400401.

Κώδικας ASCII

	Στήλη	0	1	2	3	4	5	6	7
Γραμμή	Ψηφία	0000	0001	0010	0011	0100	0101	0110	0111
0	0000	NUL	DLE	SP	0	@	P	`	p
1	0001	SOH	DC1	!	1	A	Q	a	q
2	0010	STX	DC2	"	2	B	R	b	r
3	0011	ETX	DC3	#	3	C	S	c	s
4	0100	EOT	DC4	$	4	D	T	d	t
5	0101	ENQ	NAK	%	5	E	U	e	u
6	0110	ACK	SYN	&	6	F	V	f	v
7	0111	BEL	ETB	'	7	G	W	g	w
8	1000	BS	CAN	(	8	H	X	h	x
9	1001	HT	EM	)	9	I	Y	i	y
A	1010	LF	SUB	*	:	J	Z	j	z
B	1011	VT	ESC	+	;	K	[	k	{
C	1100	FF	FS	,	<	L	\	l	|
D	1101	CR	GS	-	=	M	]	m	}
E	1110	SO	RS	.	>	N	^	n	~
F	1111	SI	US	/	?	O	_	o	DEL


ΑΡΧΗ
[D0]=[$400400]=ΔΕΔΟΜΕΝΑ
ΝΑΙ
[D0]<10;
ΟΧΙ
[D0]=[D0]+7
[D0]=[D0]+30
[$400401]=[D0]
ΤΕΛΟΣ

Διαγράμματα Ροής - Flowcharts

Το διάγραμμα ροής είναι μια διαγραμματική αναπαράσταση ενός αλγορίθμου. Το διάγραμμα ροής είναι πολύ χρήσιμο στο να γράφετε προγράμματα και να εξηγείτε αυτά σε άλλους.

Σύμβολα που χρησιμοποιούνται στο διάγραμμα ροής

Διαφορετικά σύμβολα χρησιμοποιούνται για διαφορετικές καταστάσεις στα διαγράμματα ροής, για παράδειγμα:

Η είσοδος / έξοδος και η λήψη αποφάσεων έχουν διαφορετικά σύμβολα.

Ο παρακάτω πίνακας περιγράφει όλα τα σύμβολα που χρησιμοποιούνται για τη δημιουργία ροής.

Διαγράμματα Ροής - Flowcharts

Σύμβολο	Σκοπός		Περιγραφή
	**Flow line**	**Γραμμή ροής/βέλος**	Χρησιμοποιείται για να υποδεικνύει τη ροή της λογικής συνδέοντας τα σύμβολα.
	**Terminal (Start/Stop)**	**Τερματικό (Start / Stop)**	Χρησιμοποιείται για να αντιπροσωπεύει την αρχή και το τέλος του διαγράμματος ροής.
	**Input / Output**	**Είσοδος / Έξοδος**	Χρησιμοποιείται για λειτουργία εισόδου και εξόδου.
	**Processing**	**Στάδια Επεξεργασίας**	Χρησιμοποιείται για αριθμητικές πράξεις και χειρισμούς δεδομένων.
	**Decision**	**Υποθέσεις/Αποφάσεις**	Χρησιμοποιείται για να αντιπροσωπεύει τη λειτουργία στην οποία υπάρχουν δύο εναλλακτικές λύσεις, αληθείς και ψευδείς.
	**On-page Connector**	**Σύνδεσμος επί της σελίδας**	Χρησιμοποιείται για να ενταχθεί σε διαφορετική γραμμή ροής
	**Off-page Connector**	**Συνδετήρας εκτός σελίδας**	Χρησιμοποιείται για τη σύνδεση τμήματος ροής σε διαφορετική σελίδα.
	**Predefined Process/ Function**	**Προκαθορισμένη διαδικασία / λειτουργία**	Χρησιμοποιείται για να αντιπροσωπεύει μια ομάδα δηλώσεων που εκτελούν μια εργασία επεξεργασίας.

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 **BSR** **RTS** **RTE** **RTI**	**JSR and BSR** branches to a subroutine. The PC is saved on the stack before loading the PC with the new value. 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
HEX             DC.B $0A
ASCII           DS.B 1


                ORG $400410
HEXASCII        MOVE.B HEX,D0
                CMPI.B #10,D0
                BCS ASCZ
                ADDQ.B  #7,D0
ASCZ            ADDI.B #$30, D0
                MOVE.B D0,ASCII
                END $400410
```

D0>10 C=0
D0<10 C=1

Πιο αναλυτικά η CMPI…

[D0] =$0F = 0000 1111
Imm=$0A = 0000 1010

Η εντολή `CMPI.B $0A,D0`

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

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

[D0] =$0F= 0000 1111

[NOT(Imm)+1]] =$F6= 1111 0110 +

RESULT =$05= 0000 0101

	T	S	INT	XNZVC
SR=	0	0	100	00000

ο SR έχει τα εξής CCR bits:
N=0 το αποτέλεσμα είναι θετικό.
Z=0 το αποτέλεσμα είναι μη μηδενικό.
V=0 γιατί δεν έχουμε υπερχείλιση.
C=0 γιατί ο D0 > Imm, θα είχαμε C=1 αν D0 < Imm.

Πιο αναλυτικά η CMPI…

[D0] =$00 = 0000 0000
Imm=$0A = 0000 1010

Η εντολή `CMPI.B $00,D0`

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

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

[D0] =$00= 0000 0000

[NOT(Imm)+1]] =$F6= 1111 0110

RESULT =$F6= 1111 0110

	T	S		INT		XNZVC
SR=	0	0	1	0	0	0 0 0 0 1 0 0 1

ο SR έχει τα εξής CCR bits:
N=1 το αποτέλεσμα είναι αρνητικό.
Z=0 το αποτέλεσμα είναι μη μηδενικό.
V=0 γιατί δεν έχουμε υπερχείλιση.
C=1 γιατί D0 < Imm και το χρειαζόμαστε αν θα θελήσουμε να ερμηνεύσουμε το αποτέλεσμα.

1111 0110 = Αρνητικός με μέτρο 000 1001 + C = 000 1010 = - (8+2) = -10

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 δεν 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 στην 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.

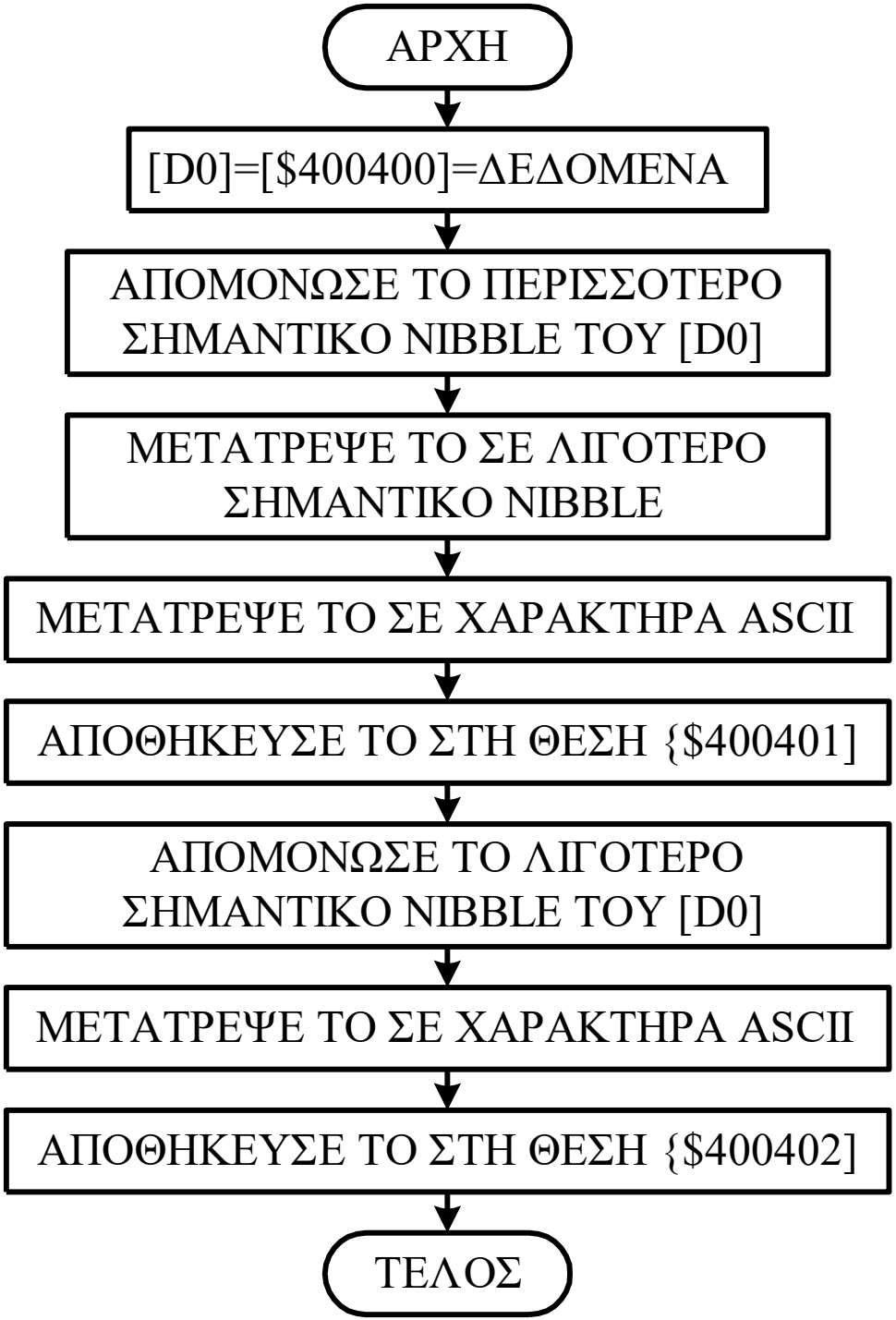
α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:
400400 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:
400410 10 39 00 40 04 00 0C 00 00 0A 65 00 00 04 5E 40
400420 06 00 00 30 13 C0 00 40 04 01 00 00 00 00 00 00

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:
400400 0A 41 00 00 00 00 00 00 00 00 00 00 00 00 00 00

ΠΑΡΑΔΕΙΓΜΑ 3.27

Να γραφτεί μια υπορουτίνα που θα μετατρέπει το δυαδικό περιεχόμενο της θέσης μνήμης $400400 σε δύο ASCII χαρακτήρες και να τους αποθηκεύει στις θέσεις μνήμης $400401 και $400402.


ΑΡΧΗ
[D0]=[$400400]=ΔΕΔΟΜΕΝΑ
ΑΠΟΜΟΝΩΣΕ ΤΟ ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ NIBBLE ΤΟΥ [D0]
ΜΕΤΑΤΡΕΨΕ ΤΟ ΣΕ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ NIBBLE
ΜΕΤΑΤΡΕΨΕ ΤΟ ΣΕ ΧΑΡΑΚΤΗΡΑ ASCII
ΑΠΟΘΗΚΕΥΣΕ ΤΟ ΣΤΗ ΘΕΣΗ {$400401]
ΑΠΟΜΟΝΩΣΕ ΤΟ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ NIBBLE ΤΟΥ [D0]
ΜΕΤΑΤΡΕΨΕ ΤΟ ΣΕ ΧΑΡΑΚΤΗΡΑ ASCII
ΑΠΟΘΗΚΕΥΣΕ ΤΟ ΣΤΗ ΘΕΣΗ {$400402]
ΤΕΛΟΣ

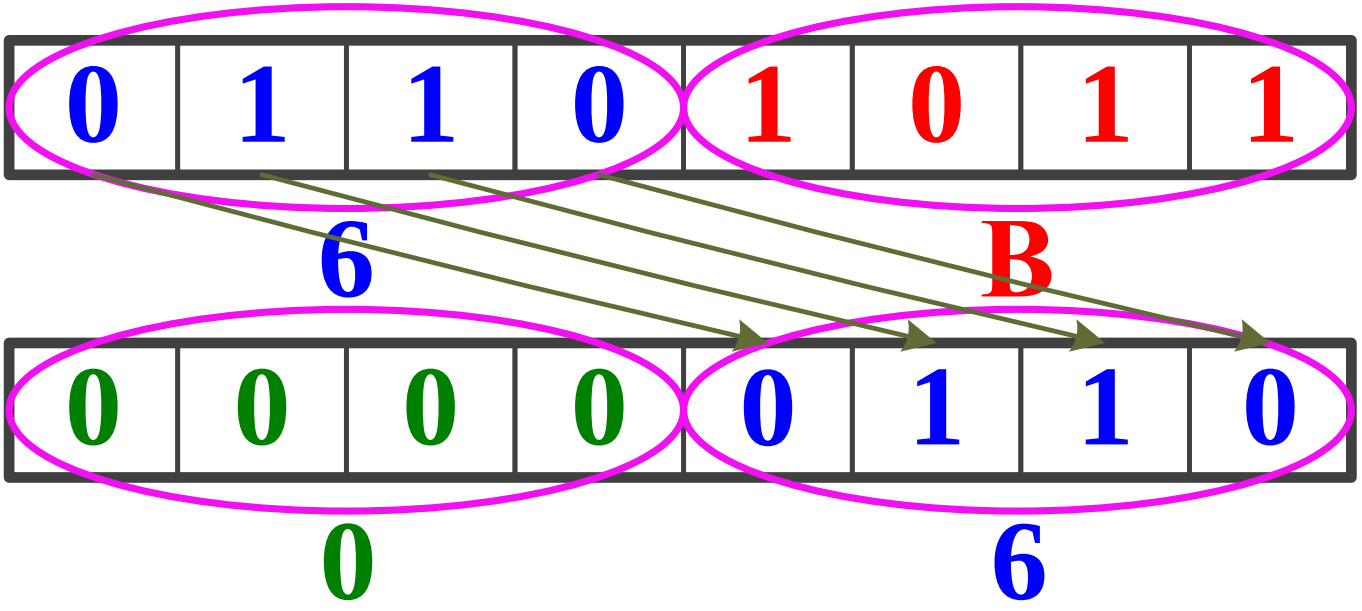
```
                ORG     $400400
BIN             DC.B    %01101011
ASCII1          DS.B    1
ASCII2          DS.B    1

                ORG     $400410
BINASCII        MOVE.B  BIN,D0
                LSR.B   #4,D0
                CMPI.B  #10,D0
                BCS     ASCZ1
                ADDQ.B  #7,D0
ASCZ1           ADDI.B  #$30,D0
                MOVE.B  D0,ASCII1
                MOVE.B  BIN,D0
                ANDI.B  #$0F,D0
                CMPI.B  #10,D0
                BCS     ASCZ2
                ADDQ.B  #7,D0
ASCZ2           ADDI.B  #$30,D0
                MOVE.B  D0,ASCII2
                END $400410
```

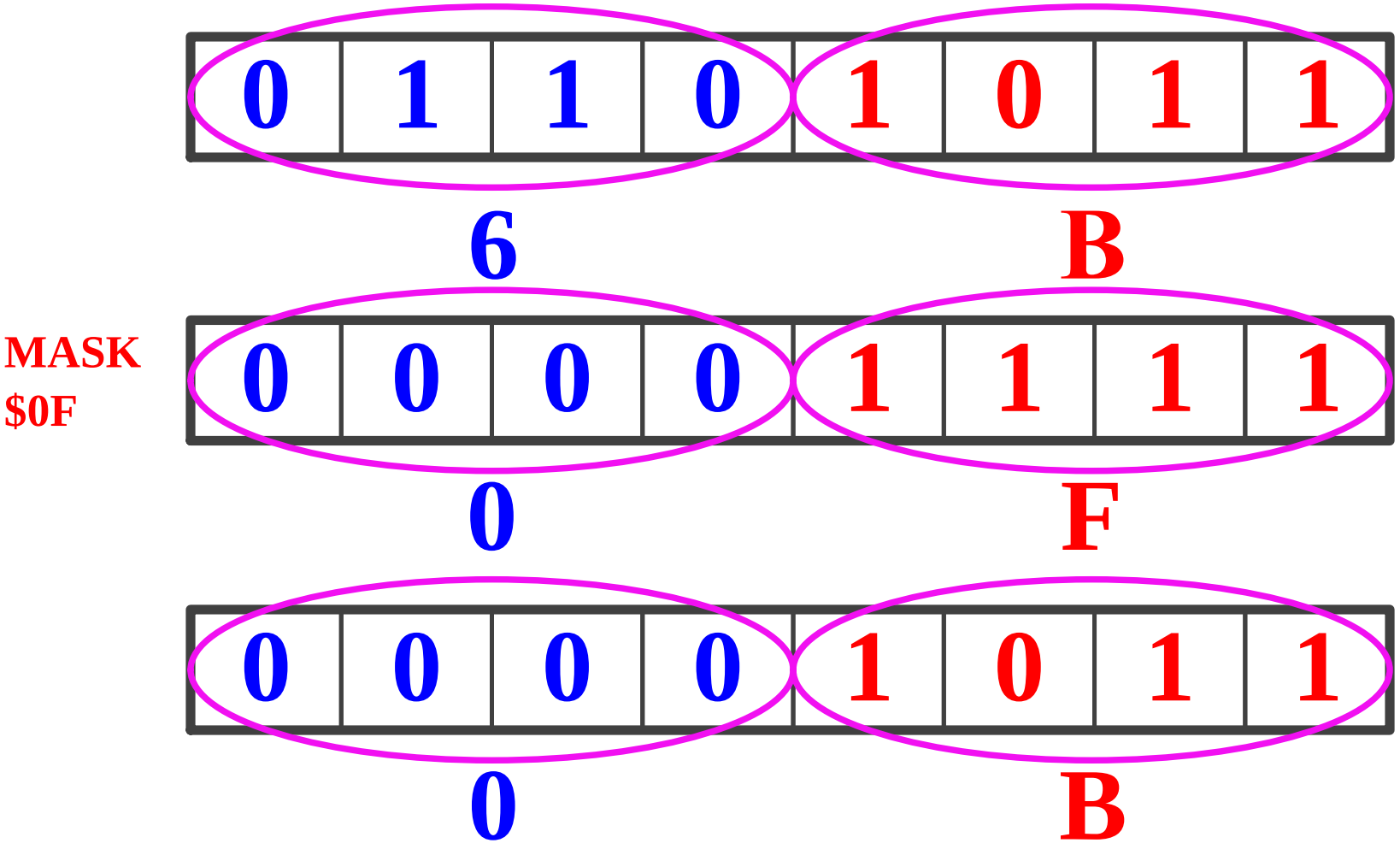
D0>10 C=0

D0<10 C=1

```
BINASCII    MOVE.B BIN,D0
            LSR.B  #4,D0
```



```
MOVE.B    BIN,D0
ANDI.B    #$0F,D0
```

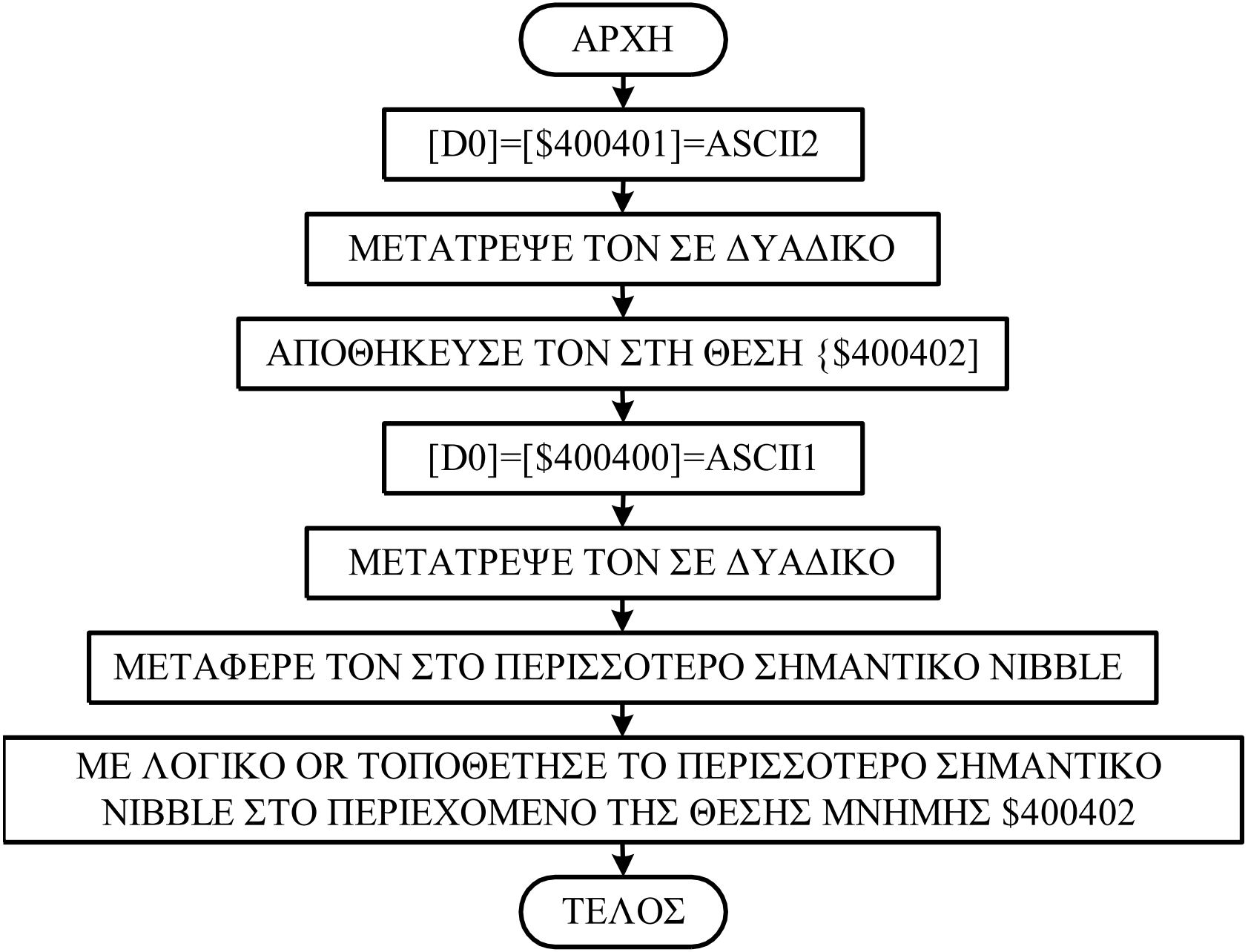


α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:
400400 6B 00 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:
400400 6B 36 42 00 00 00 00 00 00 00 00 00 00 00 00

ΠΑΡΑΔΕΙΓΜΑ 3.28

Να γραφτεί μια υπορουτίνα που θα μετατρέπει τους ASCII χαρακτήρες των θέσεων μνήμης $400400 και $400401 σε ένα δυαδικό αριθμό και να τον αποθηκεύει στη θέση μνήμης $400402.


ΑΡΧΗ
[D0]=[$400401]=ASCII2
ΜΕΤΑΤΡΕΨΕ ΤΟΝ ΣΕ ΔΥΑΔΙΚΟ
ΑΠΟΘΗΚΕΥΣΕ ΤΟΝ ΣΤΗ ΘΕΣΗ {$400402]
[D0]=[$400400]=ASCII1
ΜΕΤΑΤΡΕΨΕ ΤΟΝ ΣΕ ΔΥΑΔΙΚΟ
ΜΕΤΑΦΕΡΕ ΤΟΝ ΣΤΟ ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ NIBBLE
ΜΕ ΛΟΓΙΚΟ OR ΤΟΠΟΘΕΤΗΣΕ ΤΟ ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ NIBBLE ΣΤΟ ΠΕΡΙΕΧΟΜΕΝΟ ΤΗΣ ΘΕΣΗΣ ΜΝΗΜΗΣ $400402
ΤΕΛΟΣ

```
                ORG $400400
ASCII1          DC.B $45
ASCII2          DC.B $39
BIN             DS.B 1
                ORG $400410
ASCIIBIN        MOVE.B ASCII2,D0
                SUBI.B #$30,D0
                CMPI.B #10,D0
                BCS LNIBBLE
                SUBQ #7,D0
LNIBBLE         MOVE.B D0,BIN
                MOVE.B ASCII1,D0
                SUBI.B #$30,D0
                CMPI.B #10,D0
                BCS HNIBBLE
                SUBQ #7,D0
HNIBBLE         LSL.B #4,D0
                OR.B D0,BIN
                END $400410
```

D0>10 C=0

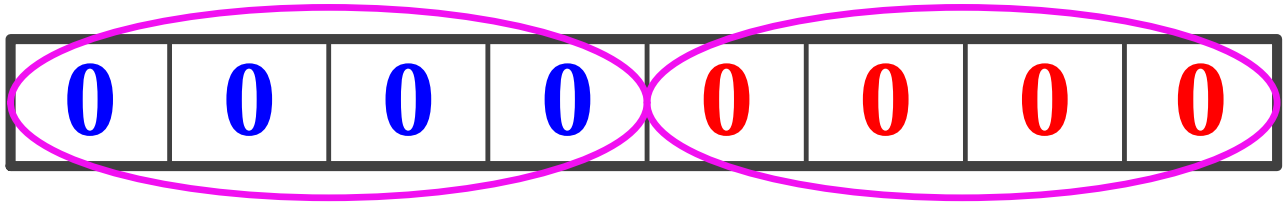
D0<10 C=1

$15 - $7 = $E

LNIBBLE **MOVE.B D0,BIN**

Αρχικά

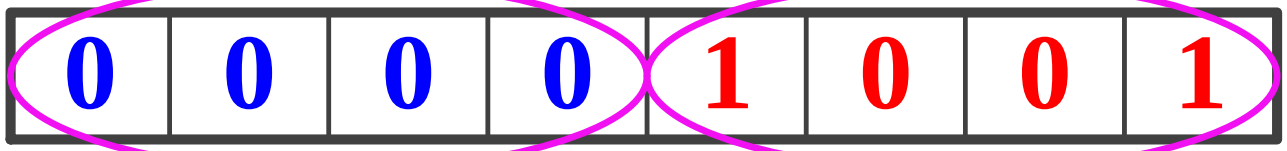
D0



0 0

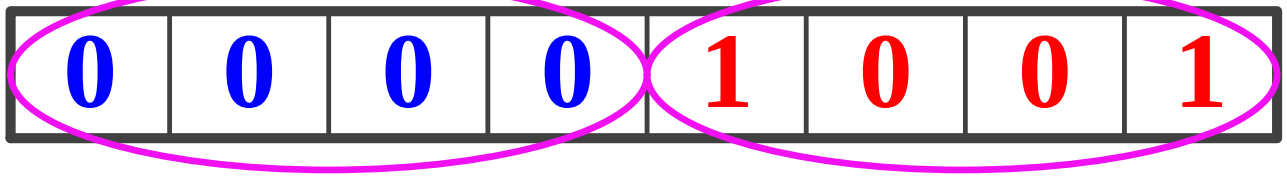
Μετά

D0

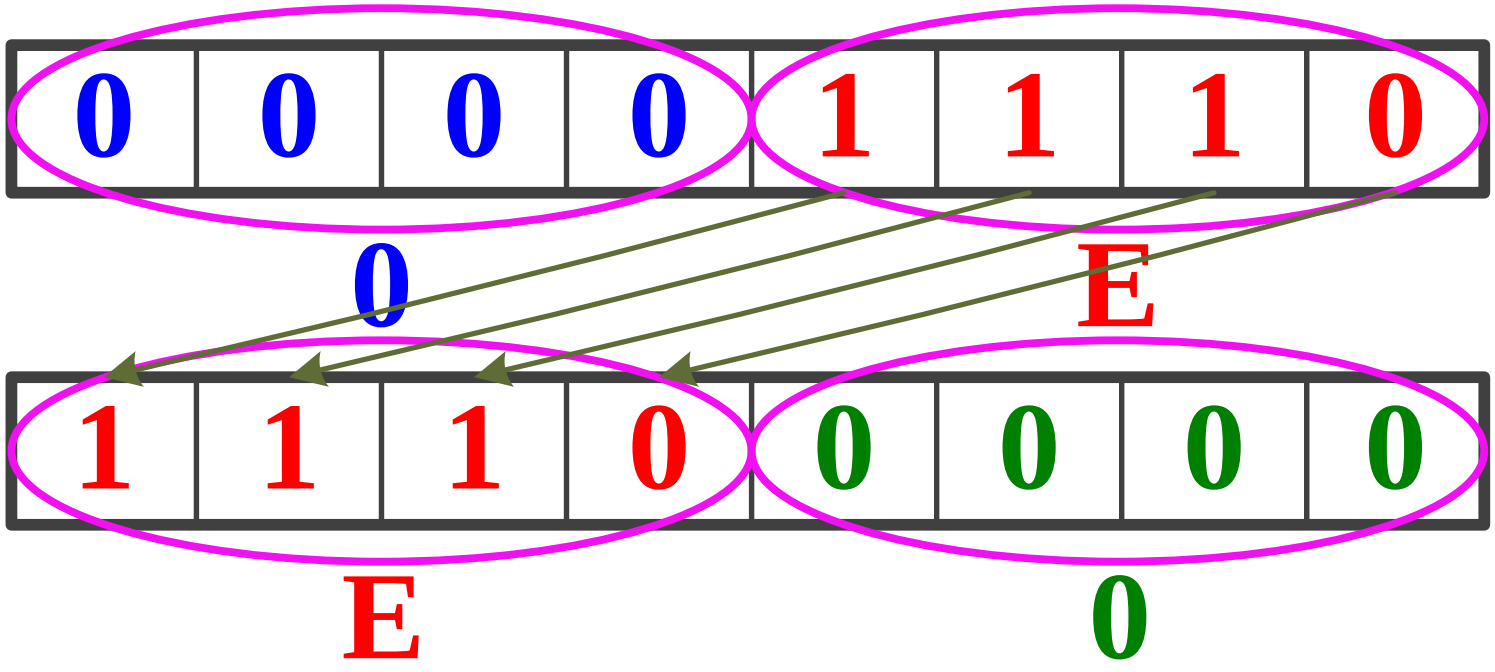


0 9

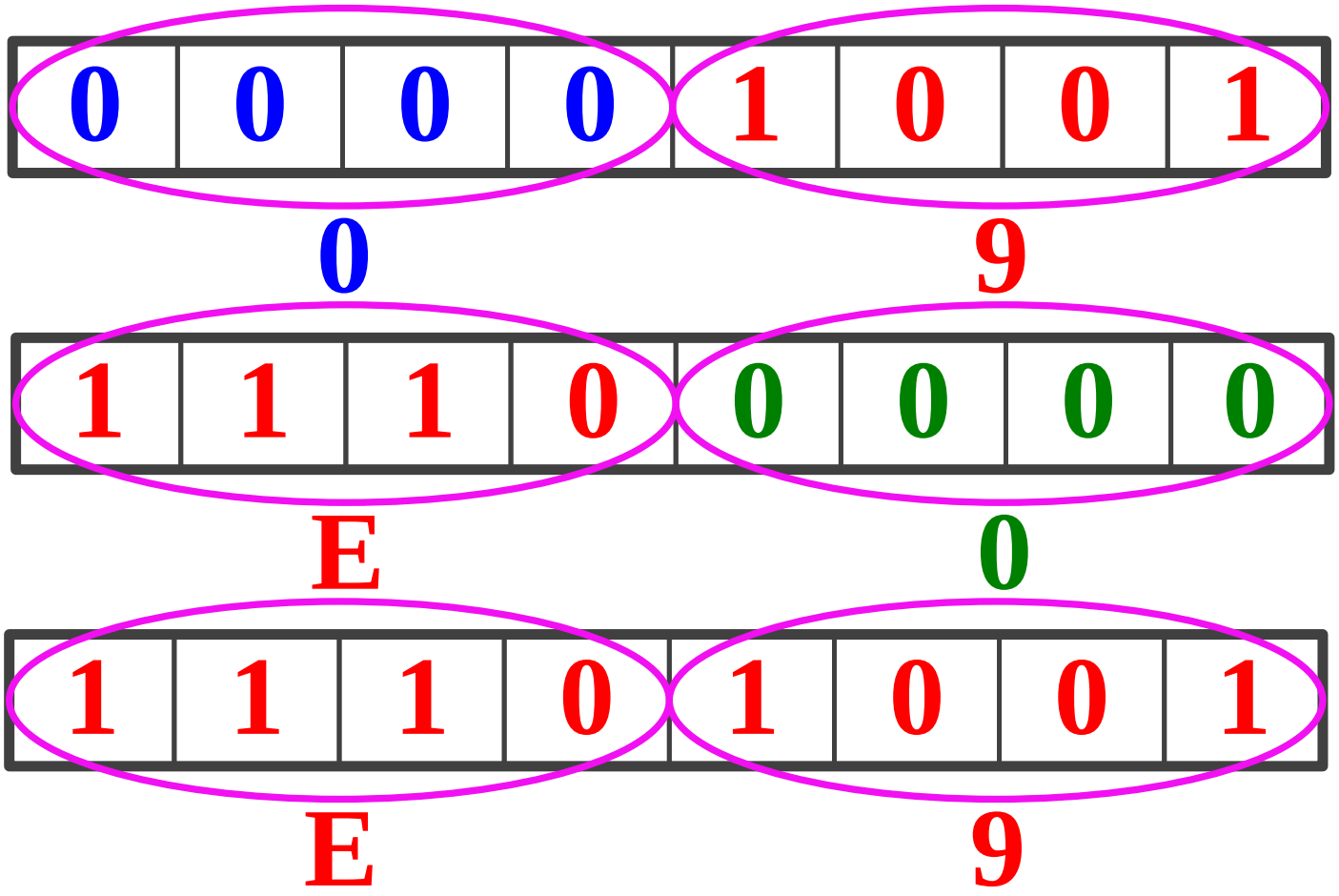
BIN



LSL.B #4,D0



OR.B D0,BIN



α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 45 39 00 00 00 00 00 00 00 00 00 00 00 00 00

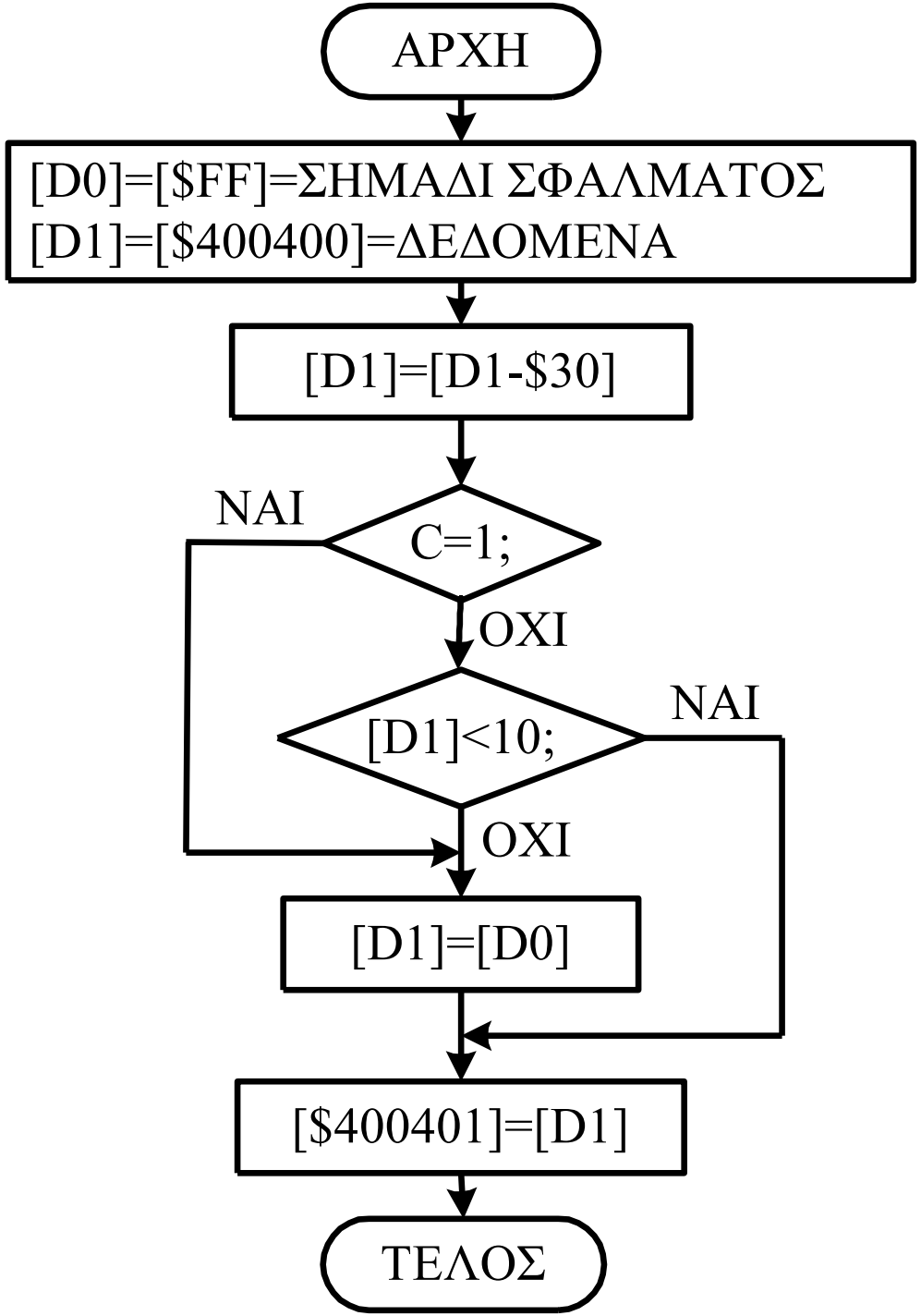
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 45 39 E9 00 00 00 00 00 00 00 00 00 00 00 00

ΠΑΡΑΔΕΙΓΜΑ 3.29

Να γραφτεί μια υπορουτίνα που θα μετατρέπει το περιεχόμενο της θέσης μνήμης $400400 από χαρακτήρα ASCII σ' ένα δεκαδικό ψηφίο και να αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400401.

Αν το περιεχόμενο της θέσης μνήμης $400400 δεν είναι αναπαράσταση ASCII ενός δεκαδικού ψηφίου να αποθηκευτεί στη θέση μνήμης $400401 ο αριθμός "FF".


ΑΡΧΗ
[D0]=[$FF]=ΣΗΜΑΔΙ ΣΦΑΛΜΑΤΟΣ
[D1]=[$400400]=ΔΕΔΟΜΕΝΑ
[D1]=[D1-$30]
NAI
C=1;
ΟΧΙ
[D1]<10;
NAI
ΟΧΙ
[D1]=[D0]
[$400401]=[D1]
ΤΕΛΟΣ

```
                ORG $400400
ASCII           DC.B $35
DEC             DS.B 1

                ORG $400410
ASCIIDEC        MOVE.B #$FF,D0
                MOVE.B ASCII,D1
                SUBI.B #$30,D1
                BCS NODECIM
                CMPI.B #10,D1
                BCS DECIMAL
NODECIM         MOVE.B D0,D1
DECIMAL         MOVE.B D1,DEC
                END $400410
```

D1>$30 C=0
D1<$30 C=1

D1>10 C=0
D1<10 C=1

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 35 00 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:

400410 10 3C 00 FF 12 39 00 40 04 00 04 01 00 30 0C 00

400420 00 0A 65 00 00 0A 0C 01 00 10 65 00 00 04 12 00

400430 13 C1 00 40 04 01 00 00 00 00 00 00 00 00 00 00

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 35 05 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Παράδειγμα 3.31

Να γραφτεί μια υπορουτίνα που θα μετατρέπει δύο ψηφία BCD που βρίσκονται στις θέσεις μνήμης $400400 και $400401 σ' έναν δυαδικό αριθμό και θα τον τοποθετεί στη θέση μνήμης $400402.

Το περισσότερο σημαντικό ψηφίο BCD βρίσκεται στη θέση μνήμης $400400.

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 $375_{10}=101101111_2=0011\ 0111\ 0101_{BCD8421}$

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

BCD σε Δυαδικό

Μία μέθοδος μετατροπής ενός αριθμού δοσμένου σε κώδικα BCD σε έναν δυαδικό αριθμό μπορεί να γίνει με τη χρήση αθροιστών, όταν έχουμε υπόψη μας τα παρακάτω:

1. Η τιμή ή το βάρος κάθε δυαδικού ψηφίου στον κώδικα BCD παριστάνεται από ένα δυαδικό αριθμό.

2. Όλες οι δυαδικές παραστάσεις βαρών ή ψηφίων που είναι "1" στον κώδικα BCD προστίθενται.

3. Το αποτέλεσμα αυτής της πρόσθεσης είναι ένας δυαδικός αριθμός ίσος με τον αντίστοιχο BCD.

Γενικά μπορεί να λεχθεί ότι:

Οι δυαδικοί αριθμοί που παριστάνουν τα βάρη των δυαδικών ψηφίων προστίθενται και ο αριθμός που προκύπτει είναι ο συνολικός δυαδικός αριθμός.

Για παράδειγμα, είναι γνωστό ότι ο δεκαδικός αριθμός 87 μπορεί να παρασταθεί σε μορφή BCD ως $10000111_{8421BCD}$.

Η ομάδα των τεσσάρων περισσότερο σημαντικών ψηφίων παριστάνουν το 80 και η ομάδα των τεσσάρων λιγότερο σημαντικών ψηφίων το 7.

Αυτό σημαίνει ότι τα τέσσερα περισσότερο σημαντικά ψηφία, που ονομάζονται και υψηλό Nybble, έχουν βάρος δεκάδες και τα τέσσερα λιγότερο σημαντικά ψηφία, που ονομάζονται και χαμηλό Nybble, έχουν βάρος μονάδες.

Για την ακρίβεια μέσα σε κάθε ομάδα το δυαδικό βάρος κάθε ψηφίου δίνεται με τον τρόπο που φαίνεται στον πίνακα.

Πίνακας 4.37

	Ψηφία δεκάδων				**Ψηφία μονάδων**			
Βάρος:	80	40	20	10	8	4	2	1
Ονομασία ψηφίου:	B_3	B_2	B_1	B_0	A_3	A_2	A_1	A_0

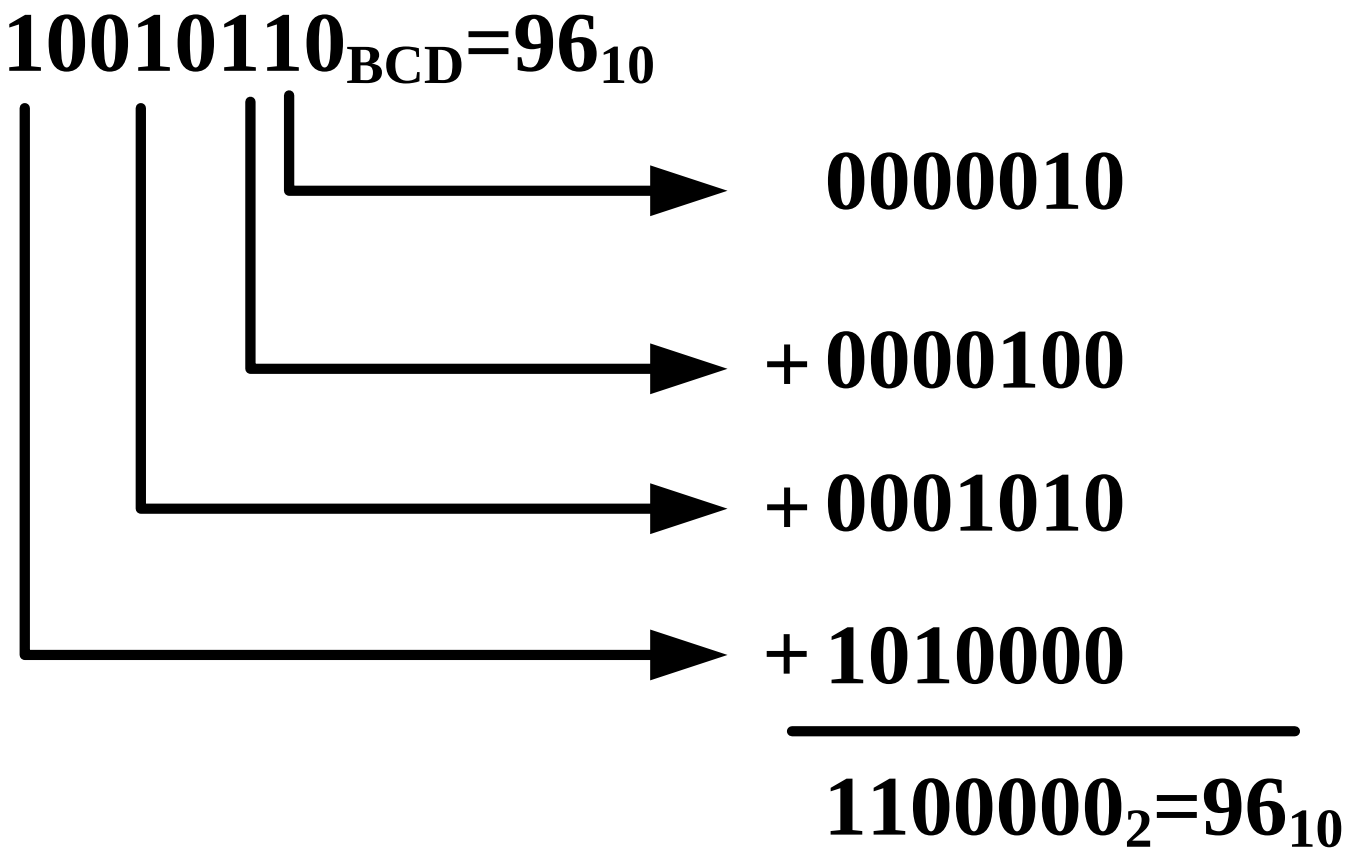
Το δυαδικό ισοδύναμο κάθε ψηφίου BCD είναι ένας δυαδικός αριθμός που παριστάνει το βάρος αυτού του ψηφίου στο πλαίσιο του αριθμό BCD, σύμφωνα με τον πίνακα.

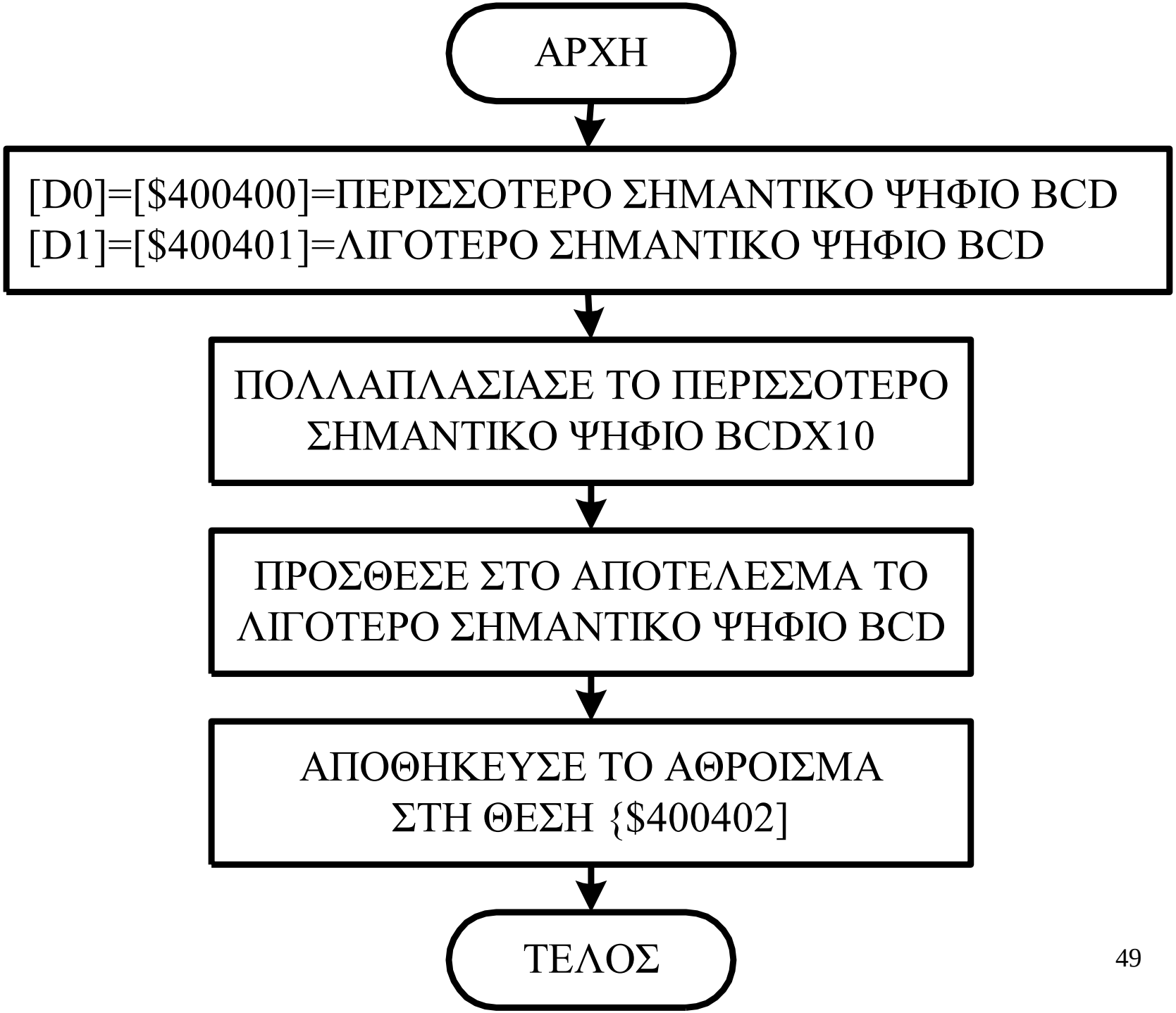
Πίνακας

		(MSB) Δυαδική παράσταση LSB)						
Ψηφίο BCD	**Βάρος BCD**	**64**	**32**	**16**	**8**	**4**	**2**	**1**
A_0	**1**	0	0	0	0	0	0	1
A_1	**2**	0	0	0	0	0	1	0
A_2	**4**	0	0	0	0	1	0	0
A_3	**8**	0	0	0	1	0	0	0
B_0	**10**	0	0	0	1	0	1	0
B_1	**20**	0	0	1	0	1	0	0
B_2	**40**	0	1	0	1	0	0	0
B_3	**80**	1	0	1	0	0	0	0

Αν προστεθούν οι δυαδικές παραστάσεις των βαρών όλων των "1" στον αριθμό BCD, τότε το αποτέλεσμα είναι ο αντίστοιχος δυαδικός αριθμός.

Για παράδειγμα ο αριθμός 10010110_{BCD} είναι ο δεκαδικός 96 και αν προστεθούν τα βάρη των ψηφίων που είναι "1", τότε το αποτέλεσμα θα είναι ο δυαδικός αριθμός 1100000 που είναι και πάλι ο δεκαδικός 96:




ΑΡΧΗ
[D0]=[$400400]=ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ ΨΗΦΙΟ BCD
[D1]=[$400401]=ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ ΨΗΦΙΟ BCD
ΠΟΛΛΑΠΛΑΣΙΑΣΕ ΤΟ ΠΕΡΙΣΣΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ ΨΗΦΙΟ BCDX10
ΠΡΟΣΘΕΣΕ ΣΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΟ ΛΙΓΟΤΕΡΟ ΣΗΜΑΝΤΙΚΟ ΨΗΦΙΟ BCD
ΑΠΟΘΗΚΕΥΣΕ ΤΟ ΑΘΡΟΙΣΜΑ ΣΤΗ ΘΕΣΗ {$400402]
ΤΕΛΟΣ

1ος Τρόπος

```
            ORG $400400
BCD1        DC.B 5
BCD2        DC.B 9
BIN         DS.B  1

            ORG       $400410
BCDBIN      MOVE.B  BCD1,D0
            MOVE.B  BCD2,D1
            MOVEQ   #10,D2
            MULU      D2,D0
            ADD.B     D0,D1
            MOVE.B  D1,BIN
            END $400410
```

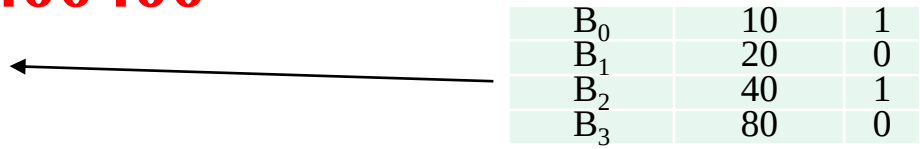
α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:
400400 05 09 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:
400410 10 39 00 40 04 00 12 39 00 40 04 01 C0 FC 00 0A
400420 D2 00 13 C1 00 40 04 02 00 00 00 00 00 00 00 00

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:
400400 05 09 3B 00 00 00 00 00 00 00 00 00 00 00 00 00

2ο Τρόπος

```
        ORG $400400
BCD1    DC.B 5
BCD2    DC.B 9
BIN     DS.B  1
```



B_0	10	1
B_1	20	0
B_2	40	1
B_3	80	0

```
        ORG      $400410
BCDBIN  MOVE.B BCD1,D0      0000 0101 = 5
        MOVE.B BCD2,D1
        LSL.B    #1,D0      0000 1010 = 10
        MOVE.B  D0,D2
        LSL.B    #2,D0      0010 1000 = 40
        ADD.B    D0,D2
        ADD.B    D2,D1
        MOVE.B  D1,BIN
        END $400410
```

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 05 09 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:

400410 10 39 00 40 04 00 12 39 00 40 04 01 C0 FC 00 0A

400420 D2 00 13 C1 00 40 04 02 00 00 00 00 00 00 00 00

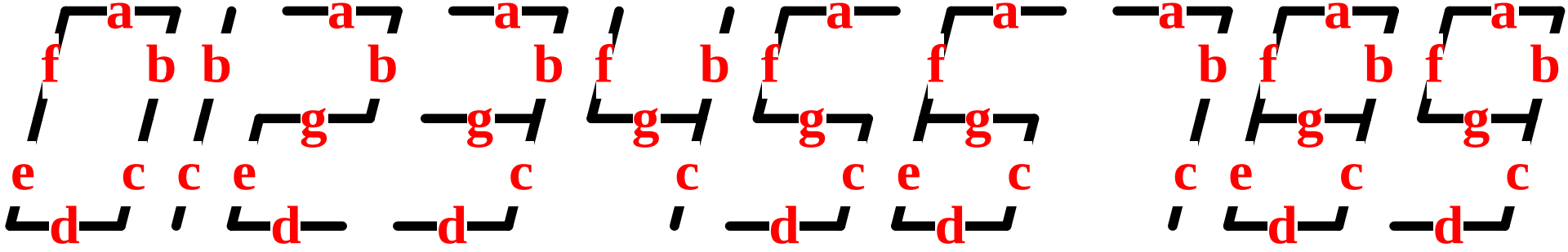
γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 05 09 3B 00 00 00 00 00 00 00 00 00 00 00 00 00

Παράδειγμα 3.33

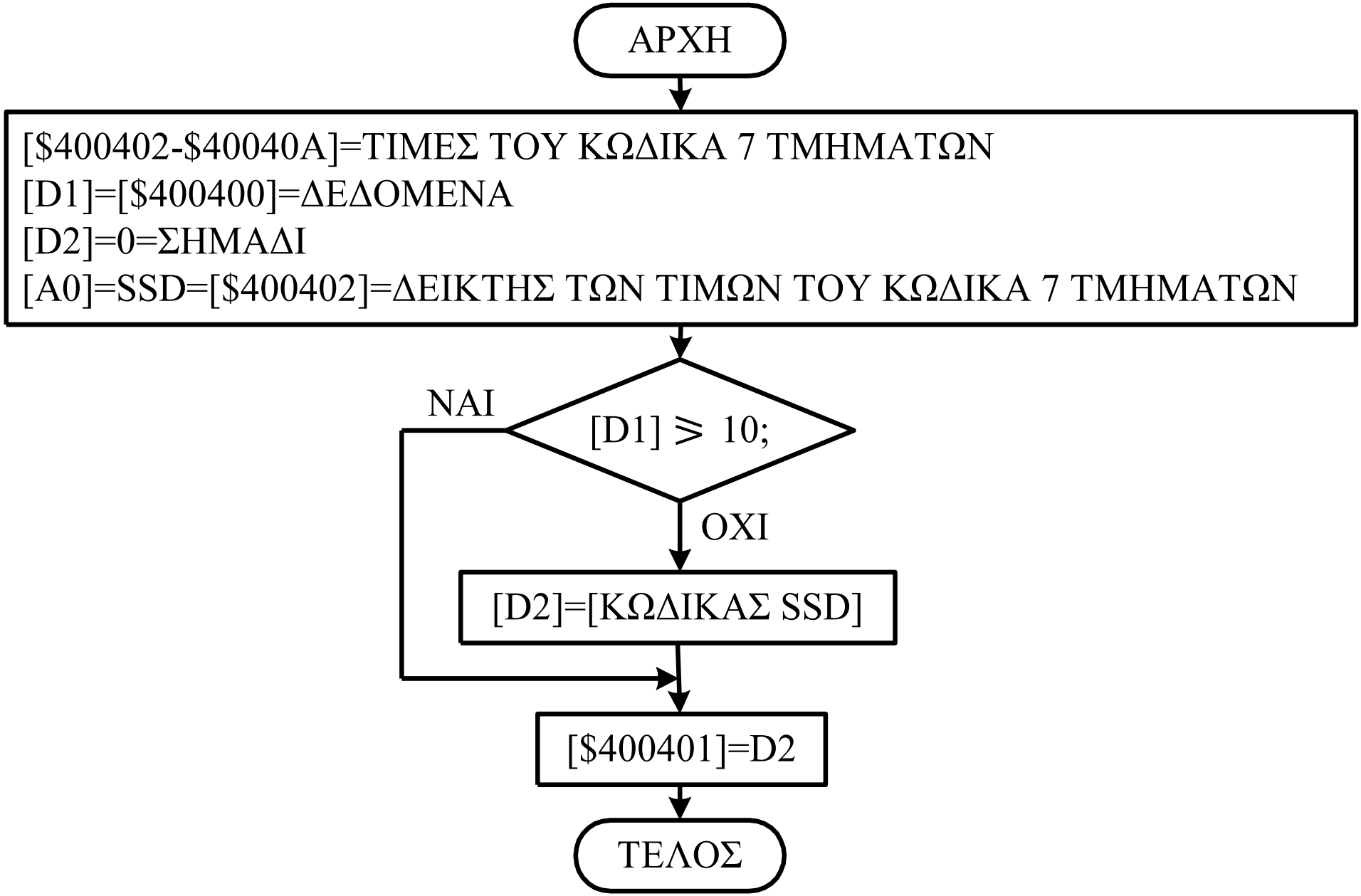
Να γραφτεί μια υπορουτίνα που θα μετατρέπει το BCD ψηφίο που βρίσκεται στη θέση μνήμης $400400 στον αντίστοιχο κώδικα 7 τμημάτων και να τον τοποθετεί στη θέση μνήμης $400401.

Αν η θέση μνήμης δεν περιέχει ψηφίο μεταξύ 0 έως 9, η θέση μνήμης $400401 να ενημερώνεται με τον αριθμό 0.



Common Cathode

Δεκ.	a	b	c	d	e	f	g	Hex
0	1	1	1	1	1	1	0	7E
1	0	1	1	0	0	0	0	30
2	1	1	0	1	1	0	1	6D
3	1	1	1	1	0	0	1	79
4	0	1	1	0	0	1	1	33
5	1	0	1	1	1	0	1	5D
6	1	0	1	1	1	1	1	5F
7	1	1	1	0	0	0	0	70
8	1	1	1	1	1	1	1	7F
9	1	1	1	1	0	1	1	7B


ΑΡΧΗ
[$400402-$40040A]=ΤΙΜΕΣ ΤΟΥ ΚΩΔΙΚΑ 7 ΤΜΗΜΑΤΩΝ
[D1]=[$400400]=ΔΕΔΟΜΕΝΑ
[D2]=0=ΣΗΜΑΔΙ
[A0]=SSD=[$400402]=ΔΕΙΚΤΗΣ ΤΩΝ ΤΙΜΩΝ ΤΟΥ ΚΩΔΙΚΑ 7 ΤΜΗΜΑΤΩΝ
ΝΑΙ
[D1] ⩾ 10;
ΟΧΙ
[D2]=[ΚΩΔΙΚΑΣ SSD]
[$400401]=D2
ΤΕΛΟΣ

```
          ORG $400400
NUM       DC.B 7
RESULT    DS.B  1
SSD       DC.B $7E,$30,$6D,$79,$33,
          $5D,$5F,$70,$7F,$7B

          ORG       $400410
BCDSSD    MOVE.B #00,D2
          MOVE.B NUM,D1
          LEA       SSD,A0
          CMPI.B   #10,D1
          BCC       NOSSD
          MOVE.B  (A0,D1),D2
NOSSD     MOVE.B  D2,RESULT
          END $400410
```

D1>10 C=0
D1<10 C=1

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:
400400 07 00 7E 30 6D 79 33 5D 5F 70 7F 7B 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:
400410 12 39 00 40 04 00 41 F9 00 40 04 02 0C 01 00 0A
400420 64 00 00 06 14 30 10 00 13 C2 00 40 04 01 00 00

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:
400400 07 70 7E 30 6D 79 33 5D 5F 70 7F 7B 00 00 00 00

ΥΠΟΡΟΥΤΙΝΕΣ ΔΙΕΡΕΥΝΗΣΗΣ ΚΑΙ ΣΥΓΚΡΙΣΗΣ ΠΙΝΑΚΩΝ ΚΑΙ ΟΡΜΑΘΩΝ

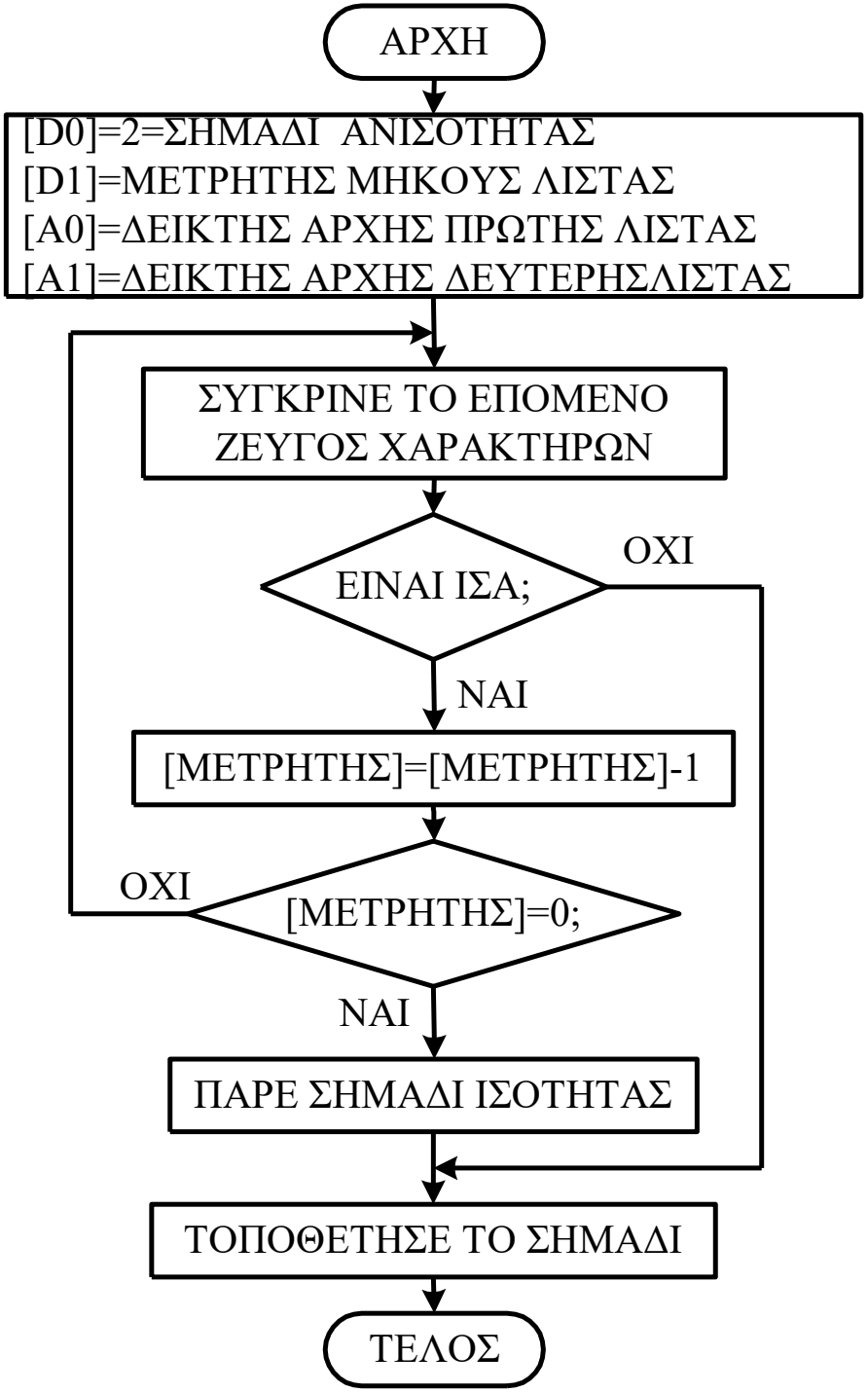
Παράδειγμα 3.34

Να γραφτεί μια υπορουτίνα που θα συγκρίνει 2 σειρές χαρακτήρων ASCII για να ελέγξει αν είναι ίδιες.

Το μήκος των σειρών βρίσκεται αποθηκευμένο στη θέση μνήμης $400401.

Οι σειρές αρχίζουν από τις θέσεις μνήμης $400403 η μία και $400413 η άλλη.

Αν οι δύο σειρές είναι ίδιες να τοποθετεί στη θέση μνήμης $400402 το αριθμό "1" διαφορετικά το "2".


ΑΡΧΗ
[D0]=2=ΣΗΜΑΔΙ ΑΝΙΣΟΤΗΤΑΣ
[D1]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[A0]=ΔΕΙΚΤΗΣ ΑΡΧΗΣ ΠΡΩΤΗΣ ΛΙΣΤΑΣ
[A1]=ΔΕΙΚΤΗΣ ΑΡΧΗΣ ΔΕΥΤΕΡΗΣΛΙΣΤΑΣ
ΣΥΓΚΡΙΝΕ ΤΟ ΕΠΟΜΕΝΟ ΖΕΥΓΟΣ ΧΑΡΑΚΤΗΡΩΝ
ΕΙΝΑΙ ΙΣΑ;
ΟΧΙ
ΝΑΙ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
[ΜΕΤΡΗΤΗΣ]=0;
ΟΧΙ
ΝΑΙ
ΠΑΡΕ ΣΗΜΑΔΙ ΙΣΟΤΗΤΑΣ
ΤΟΠΟΘΕΤΗΣΕ ΤΟ ΣΗΜΑΔΙ
ΤΕΛΟΣ

```
                ORG $400400
LENGTH          DC.B 9
SHMADI          DS.B 1
NUMS1           DC.B $50,$4F,$47,$41,$52,$49,$44,$49,$53
NUMS2           DC.B $50,$4F,$47,$41,$52,$49,$44,$49,$53

                ORG     $400420
CMPLST          MOVEQ  #2,D0
                CLR.L   D1
                MOVE.B  LENGTH,D1
                LEA     NUMS1,A0
                LEA     NUMS2,A1
LOOP            MOVE.B  (A0)+,D2
                CMP.B   (A1)+,D2
                BNE     FIN
                SUBQ.B  #1,D1
                BNE      LOOP
                SUBQ.B   #1,D0
FIN             MOVE.B   D0,SHMADI
                END $400410
```

Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 09 01 50 4F 47 41 52 49 44 49 53 50 4F 47 41 52

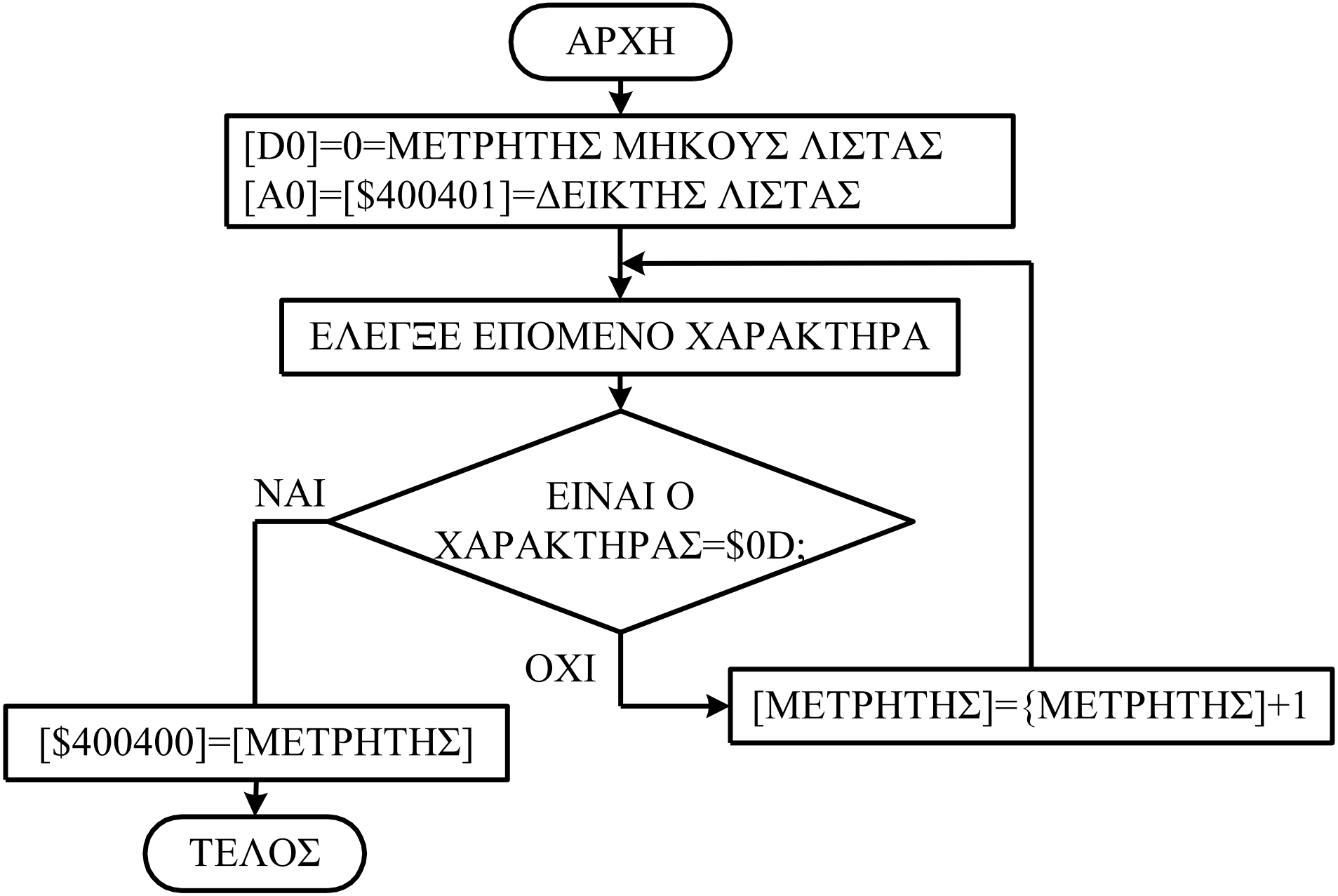
400410 49 44 49 53 00 00 00 00 00 00 00 00 00 00 00 00

Παράδειγμα 3.35

Να γραφτεί μια υπορουτίνα που θα υπολογίζει το μήκος μιας σειράς χαρακτήρων ASCII (επτά ψηφία με το περισσότερο σημαντικό ψηφίο μηδέν).

Η σειρά αρχίζει από τη θέση μνήμης $400401 και το τέλος της σημαδεύεται από τον χαρακτήρα CARRIEGE RETURN ("CR"=$0D).

Το μήκος της σειράς (εκτός από τον χαρακτήρα "CR") να τοποθετηθεί στη θέση μνήμης $400400.


ΑΡΧΗ
[D0]=0=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[A0]=[$400401]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
ΕΛΕΓΞΕ ΕΠΟΜΕΝΟ ΧΑΡΑΚΤΗΡΑ
ΕΙΝΑΙ Ο ΧΑΡΑΚΤΗΡΑΣ=$0D;
ΝΑΙ
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]={ΜΕΤΡΗΤΗΣ]+1
[$400400]=[ΜΕΤΡΗΤΗΣ]
ΤΕΛΟΣ

```
                ORG $400400
LENGTH          DS.B 1
LIST            DC.B $6D,$69,$63,$72,
                $6F,$70,$72,$6F,$63,$65,
                $73,$73,$6F,$72,$20,$36,
                $38,$30,$30,$30,$0D
                ORG     $400440
ASCIILIST       CLR.L   D0
                LEA     LIST,A0
LOOP            CMPI.B  #$0D,(A0)+
                BEQ     DONE
                ADDQ.B  #1,D0
                BRA     LOOP
DONE            MOVE.B  D0,LENGTH
                END $400440
```

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 00 6D 69 63 72 6F 70 72 6F 63 65 73 73 6F 72 20

400410 36 38 30 30 30 0D 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

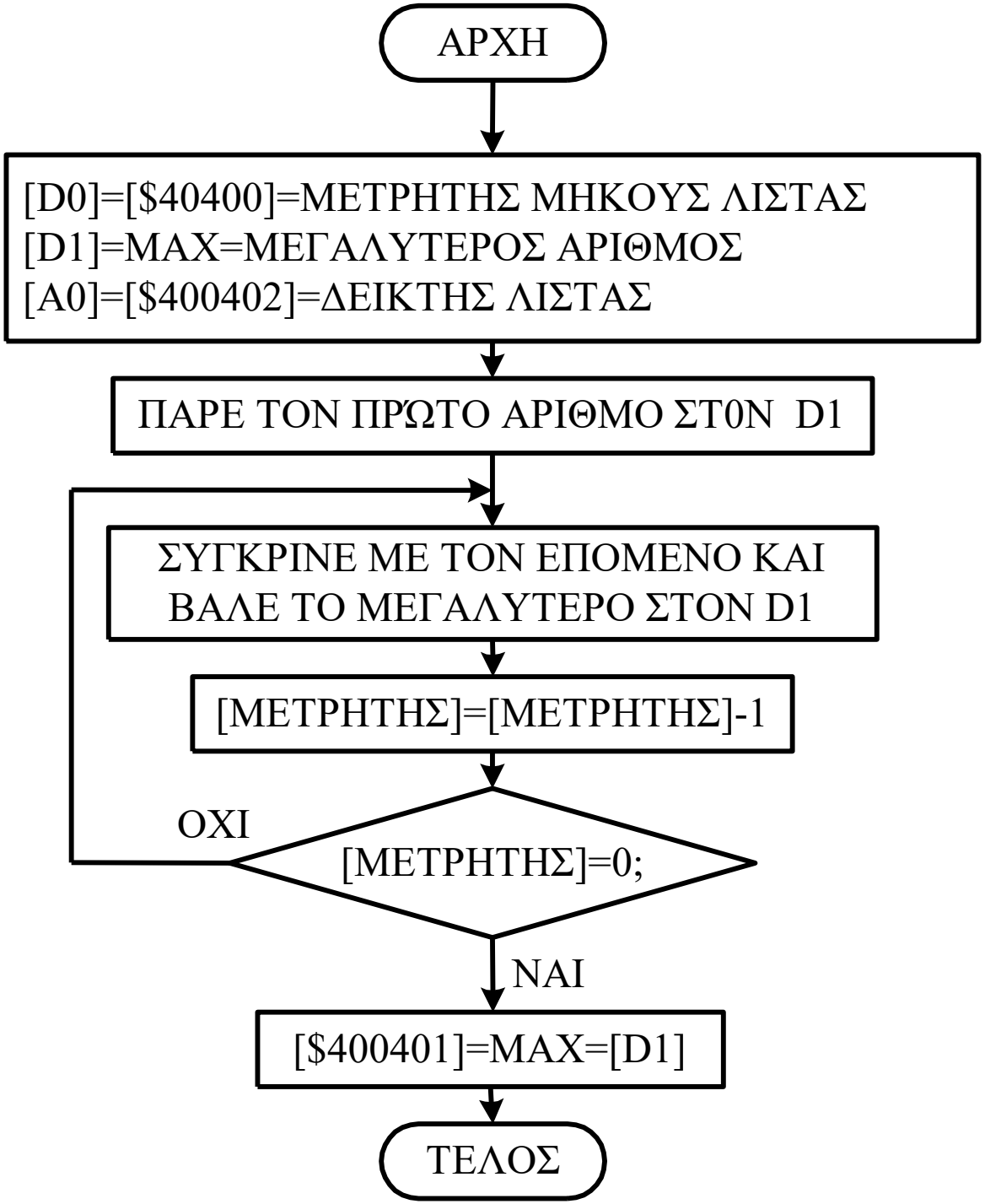
400400 14 6D 69 63 72 6F 70 72 6F 63 65 73 73 6F 72 20

400410 36 38 30 30 30 0D 00 00 00 00 00 00 00 00 00 00

Παράδειγμα 3.36

Να γραφτεί μια υπορουτίνα που θα εντοπίζει το μεγαλύτερο μη προσημασμένο αριθμό από ένα πλήθος αριθμών μήκους byte. Το πλήθος των αριθμών είναι αποθηκευμένο στη θέση μνήμης $400400 και η σειρά αρχίζει απ' τη θέση μνήμης $400402.

Το αποτέλεσμα να αποθηκευτεί στη θέση μνήμης $400401.


ΑΡΧΗ
[D0]=[$40400]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[D1]=MAX=ΜΕΓΑΛΥΤΕΡΟΣ ΑΡΙΘΜΟΣ
[A0]=[$400402]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
ΠΑΡΕ ΤΟΝ ΠΡΏΤΟ ΑΡΙΘΜΟ ΣΤ0Ν D1
ΣΥΓΚΡΙΝΕ ΜΕ ΤΟΝ ΕΠΟΜΕΝΟ ΚΑΙ
ΒΑΛΕ ΤΟ ΜΕΓΑΛΥΤΕΡΟ ΣΤΟΝ D1
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]=0;
ΝΑΙ
[$400401]=MAX=[D1]
ΤΕΛΟΣ

```
          ORG $400400
LENGTH    DC.B 5
MAX       DS.B 1
LIST      DC.B $6D,$4D,$42,$71,$12

          ORG $400410
SUBRTN    CLR.L D0
          MOVE.B LENGTH,D0
          CLR.L D1
          LEA LIST,A0
LOOP      MOVE.B (A0)+,D2
          CMP.B D2,D1
          BCC NOCHG
          MOVE.B D2,D1
NOCHG     SUBQ.B #1,D0
          BNE LOOP
          MOVE.B D1,MAX
          END $400410
```

D1>D2 C=0
D1<D2 C=1

α. Περιοχή μνήμης δεδομένων πριν απ' την εκτέλεση του προγράμματος:

400400 05 00 6D 4D 42 71 12 00 00 00 00 00 00 00 00

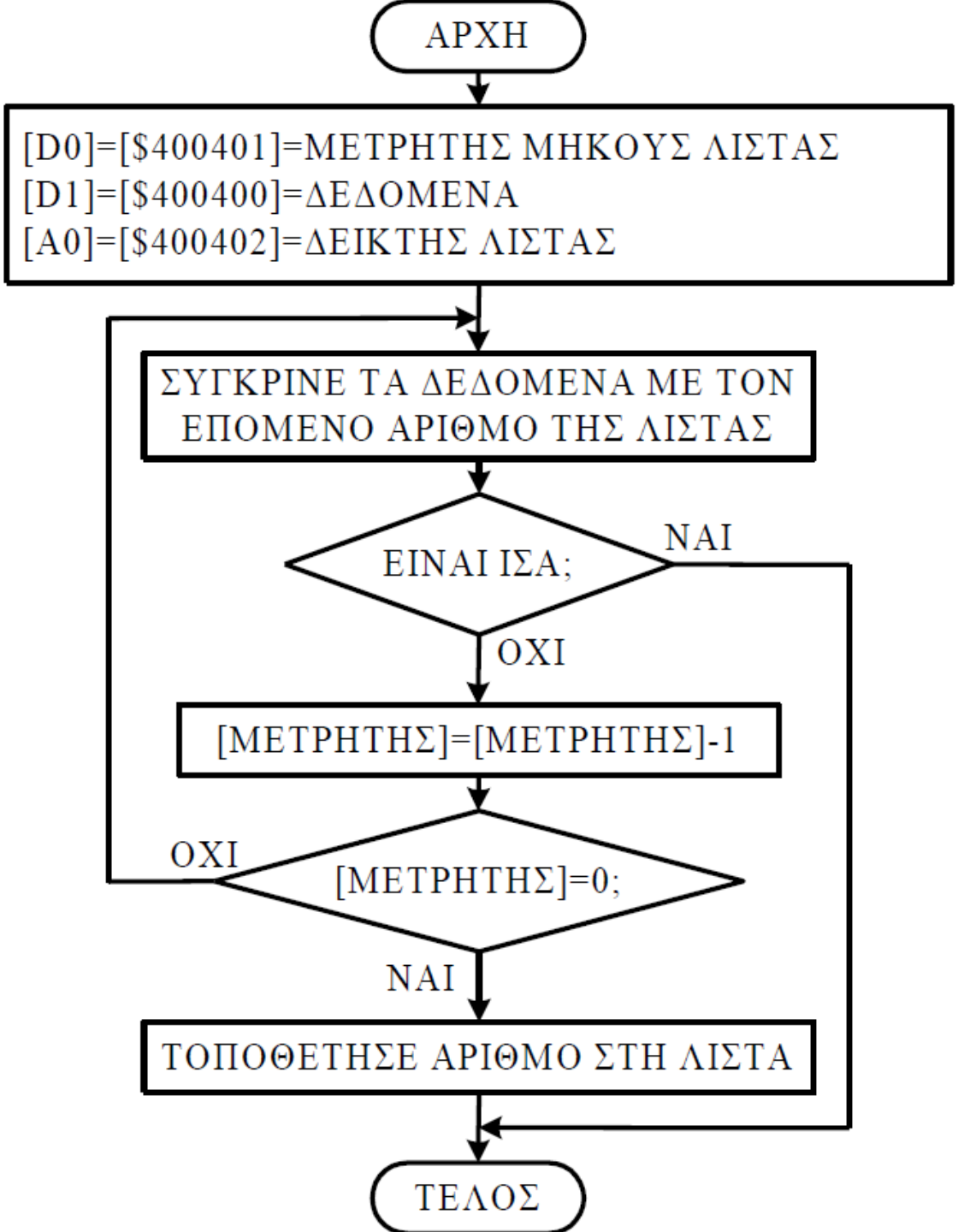
β. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 05 71 6D 4D 42 71 12 00 00 00 00 00 00 00 00

Παράδειγμα 3.37

Να γραφτεί μια υπορουτίνα που να τοποθετεί τα περιεχόμενα της θέσης μνήμης $400400 στο τέλος μιας λίστας, αν δεν είναι ήδη παρόν στη λίστα.

Το μήκος της λίστας βρίσκεται αποθηκευμένο στη θέση μνήμης $400401 και η λίστα αρχίζει από τη θέση μνήμης $400402.


ΑΡΧΗ
[D0]=[$400401]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[D1]=[$400400]=ΔΕΔΟΜΕΝΑ
[A0]=[$400402]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
ΣΥΓΚΡΙΝΕ ΤΑ ΔΕΔΟΜΕΝΑ ΜΕ ΤΟΝ ΕΠΟΜΕΝΟ ΑΡΙΘΜΟ ΤΗΣ ΛΙΣΤΑΣ
ΕΙΝΑΙ ΙΣΑ;
ΝΑΙ
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
[ΜΕΤΡΗΤΗΣ]=0;
ΟΧΙ
ΝΑΙ
ΤΟΠΟΘΕΤΗΣΕ ΑΡΙΘΜΟ ΣΤΗ ΛΙΣΤΑ
ΤΕΛΟΣ

```
                ORG $400400
NUM             DC.B $55
LENGTH          DC.B 18
LIST            DC.B $6D,$7A,$85,
                $77,$82,$13,$4D,$42,
                $35,$79,$23,$F1,$23,
                $45,$A0,$12,$34,$5E

                ORG       $400420
CHECK           MOVE.B    LENGTH,D0
                LEA       LIST,A0
                MOVE.B    NUM,D1
LOOP            CMP.B     (A0)+, D1
                BEQ       FIN
                SUBI.B     #1,D0
                BNE       LOOP
                MOVE.B    D1, (A0)
FIN             END $400420
```

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 55 12 6D 7A 85 77 82 13 4D 42 35 79 23 F1 23 45

400410 A0 12 34 5E 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

400400 55 12 6D 7A 85 77 82 13 4D 42 35 79 23 F1 23 45

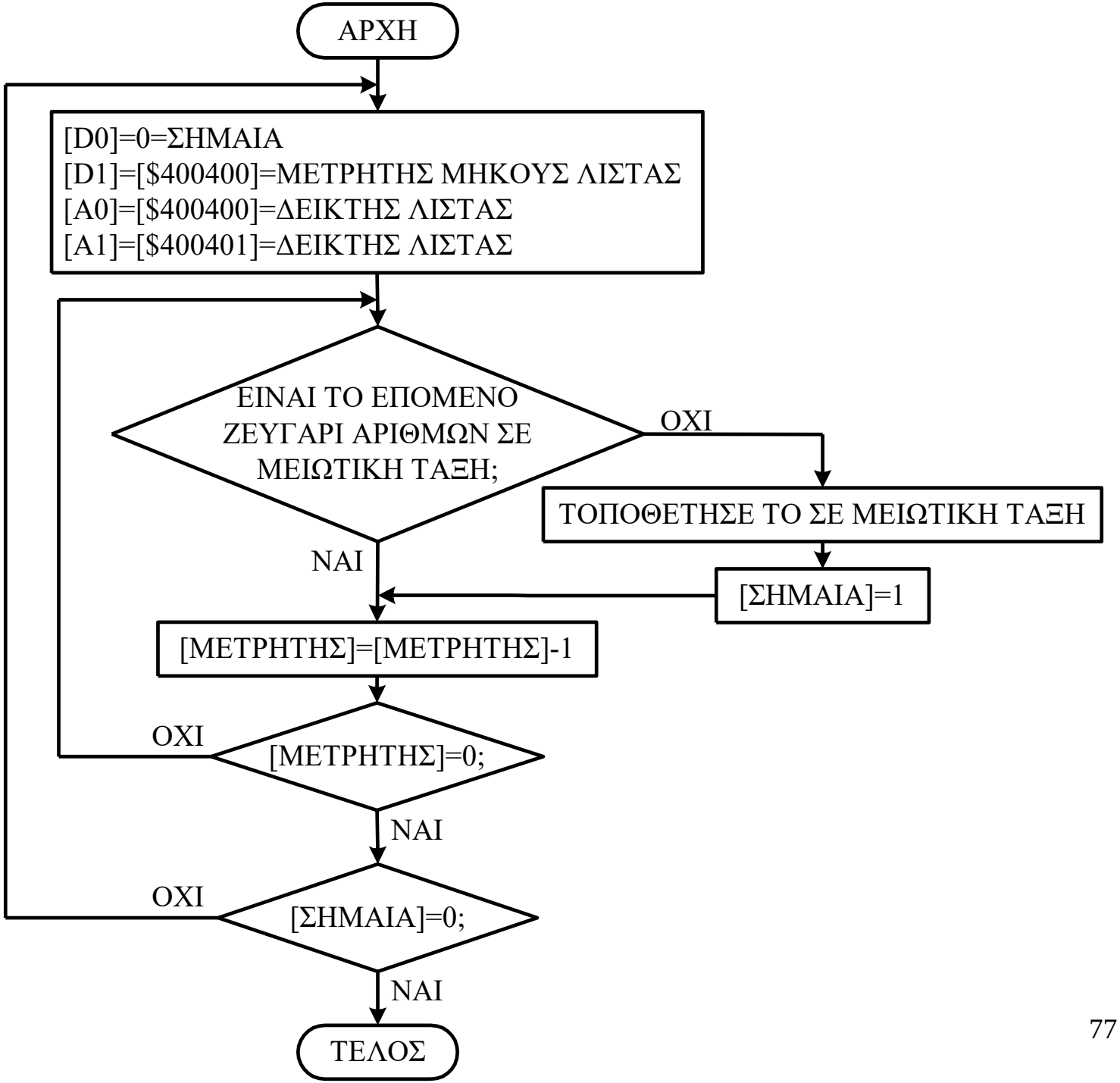
400410 A0 12 34 5E 55 00 00 00 00 00 00 00 00 00 00 00

ΥΠΟΡΟΥΤΙΝΕΣ ΤΟΠΟΘΕΤΗΣΗΣ ΣΕ ΑΥΞΗΤΙΚΗ ΚΑΙ ΜΕΙΩΤΙΚΗ ΤΑΞΗ

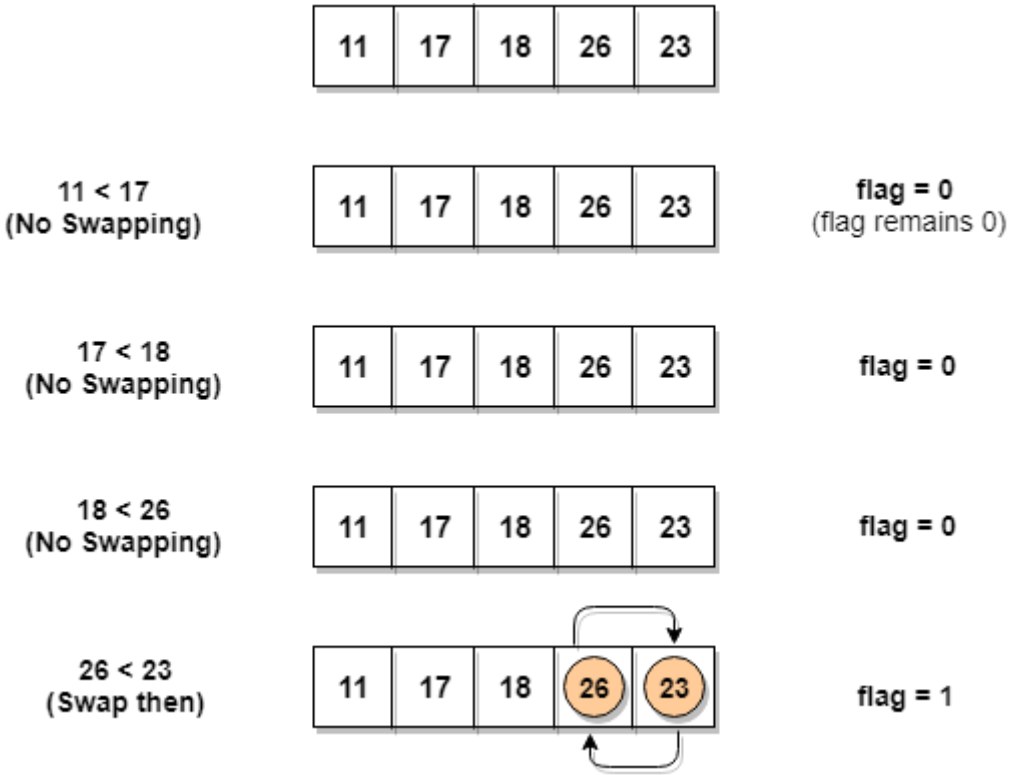
Παράδειγμα 3.38

Να γραφτεί μια υπορουτίνα που να τοποθετεί σε μειωτική τάξη μια λίστα μη προσημασμένων δυαδικών αριθμών 18 ψηφίων.

Το μήκος της λίστας βρίσκεται στη θέση μνήμης $400400 και η λίστα αρχίζει από τη θέση μνήμης $400401.


ΑΡΧΗ
[D0]=0=ΣΗΜΑΙΑ
[D1]=[$400400]=ΜΕΤΡΗΤΗΣ ΜΗΚΟΥΣ ΛΙΣΤΑΣ
[A0]=[$400400]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
[A1]=[$400401]=ΔΕΙΚΤΗΣ ΛΙΣΤΑΣ
ΕΙΝΑΙ ΤΟ ΕΠΟΜΕΝΟ ΖΕΥΓΑΡΙ ΑΡΙΘΜΩΝ ΣΕ ΜΕΙΩΤΙΚΗ ΤΑΞΗ;
ΟΧΙ
ΤΟΠΟΘΕΤΗΣΕ ΤΟ ΣΕ ΜΕΙΩΤΙΚΗ ΤΑΞΗ
[ΣΗΜΑΙΑ]=1
ΝΑΙ
[ΜΕΤΡΗΤΗΣ]=[ΜΕΤΡΗΤΗΣ]-1
ΟΧΙ
[ΜΕΤΡΗΤΗΣ]=0;
ΝΑΙ
ΟΧΙ
[ΣΗΜΑΙΑ]=0;
ΝΑΙ
ΤΕΛΟΣ

BubbleSort



Για καλύτερο έλεγχο του αλγόριθμου ταξινόμησης μπορούμε να έχουμε μια σημαία που θα γίνεται flag=1 όταν έχουμε ανταλλαγή δεδομένων κατά την εκτέλεση ενός βρόγχου.

Έτσι αν σε ένα βρόγχο έχουμε ανταλλαγή, τότε flag=1 και ο βρόγχος τρέχει μια ακόμη φορά. Αν δεν έχουμε ανταλλαγή σημαίνει ότι ολοκληρώθηκε η ταξινόμηση και τερματίζει.

```
                ORG $400400
LENGTH          DC.B 18
LIST            DC.B $6D,$7A,$85,$77,$82,$13,$4D,$42,
                $35,$79,$23,$F1,$23,$45,$A0,$12,$34,$5E

                ORG     $400420
 SORT           MOVE.B  LENGTH,D1
                SUBQ.B  #1,D1
                LEA     LIST-1,A0
                LEA     LIST,A1
LOOP            MOVE.B  (A0,D1),D2
                CMP.B   (A1,D1),D2
                BCC     COUNT
                MOVEQ   #1,D0
                MOVE.B  D2,D3
                MOVE.B  (A1,D1),(A0,D1)
                MOVE.B  D3,(A1,D1)
COUNT           SUBQ.B  #1,D1
                BNE     LOOP
                SUBQ.B  #1,D0
                BEQ     SORT
                END $400420
```

D2>(A1,D1) C=0, μειωτική
D2<(A1,D1) C=1, sort

Μειώνουμε το flag κατά ένα και αν είναι αρνητικός τερματίζει. Άρα αν έρθει από flag=1 τρέχει μια φορά ακόμη.

α. Περιοχή μνήμης δεδομένων πριν απ’ την εκτέλεση του προγράμματος:

400400 12 6D 7A 85 77 82 13 4D 42 35 79 23 F1 23 45 A0

400410 12 34 5E 00 00 00 00 00 00 00 00 00 00 00 00 00

β. Περιοχή μνήμης κώδικα προγράμματος:

400420 42 80 42 81 12 39 00 40 04 00 53 01 41 F9 00 40

400430 04 00 43 F9 00 40 04 01 14 30 10 00 B4 31 10 00

400440 64 00 00 10 70 01 16 02 11 B1 10 00 10 00 13 83

400450 10 00 53 01 66 E2 53 00 67 C6 00 00 00 00 00 00

γ. Περιοχή μνήμης δεδομένων μετά την εκτέλεση του προγράμματος:

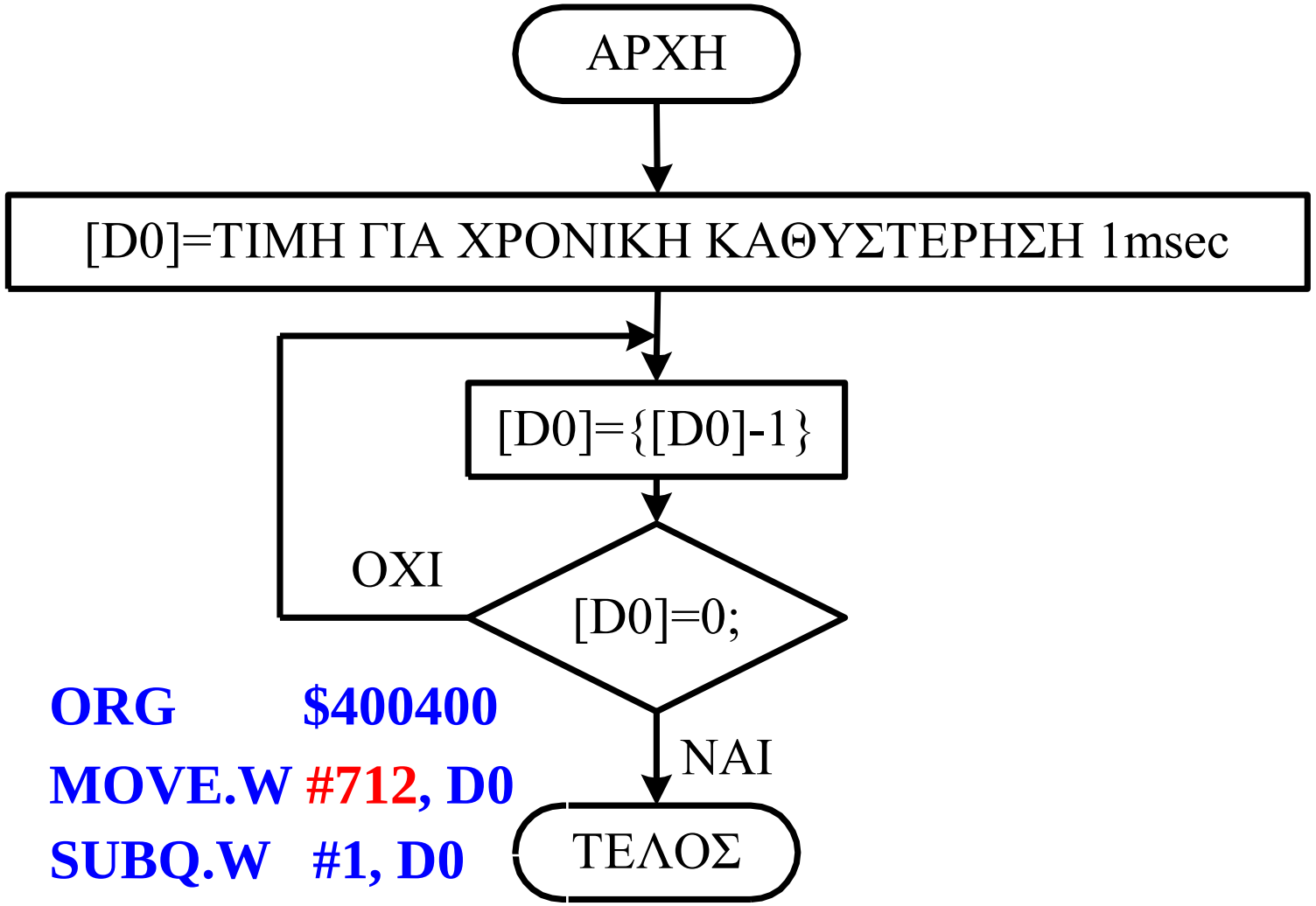
400400 12 F1 A0 85 82 7A 79 77 6D 5E 4D 45 42 35 34 23

400410 23 13 12 00 00 00 00 00 00 00 00 00 00 00 00 00

ΥΠΟΡΟΥΤΙΝΕΣ ΧΡΟΝΟΚΑΘΥΣΤΕΡΗΣΗΣ

Παράδειγμα 3.46

Να γραφτεί μια υπορουτίνα χρονοκαθυστέρησης ενός msec




```
        ORG     $400400
        MOVE.W  #712, D0
LOOP    SUBQ.W  #1, D0
        BNE     LOOP
        END $400400
```

Ο M68000 έχει ρολόι 10MHz άρα χρόνο κύκλου 1/10MHz=0.1μsec.

Η εντολή MOVE.W #XXX,Dn χρειάζεται 12 κύκλους για να εκτελεστεί και άρα 1.2μsec.

Η εντολή SUBQ.W op#,Dn χρειάζεται 4 κύκλους και άρα 0.4μsec για μέγεθος byte ή λέξης.

Η εντολή BNE χρειάζεται, όταν προκαλείται μετατόπιση, 10 κύκλους και άρα 1μsec για μετατόπιση μήκους byte ενώ όταν δεν προκαλείται μετατόπιση 8 κύκλους και άρα 0.8μsec.

Η εντολή RTS χρειάζεται 16 κύκλους και άρα 1.6μsec.

Στην προκειμένη περίπτωση θα εκτελεστούν μία φορά οι εντολές, MOVE.W #XXX,Dn, BNE LOOP, και RTS και θα χρειαστούν χρόνο 1.2+0.8+1.6=3.6μsec.

Επομένως, ο βρόχος αφαίρεσης και διακλάδωσης πρέπει να εκτελείται τόσες φορές ώστε να παράγει τα υπόλοιπα 996.4μsec.

Ο βρόχος όταν εκτελείται μια φορά προκαλεί χρονοκαθυστέρηση 0.4+1=1.4μsec.
Άρα, για να προκαλέσει χρονοκαθυστέρηση 996.4μsec πρέπει να εκτελεστεί 996.4/1.4=712 φορές.
Επομένως, ο καταχωρητής D0 πρέπει να φορτωθεί με τον αριθμό 712.

Παράδειγμα 3.47

Να γραφτεί μια υπορουτίνα χρονοκαθυστέρησης ενός δευτερολέπτου.

```
        ORG     $400400
DELAY   MOVE.L  #555553,D6
LOOP    SUBQ.L  #1,D6
        BNE     LOOP
        END $400400
```

Η υπορουτίνα αυτή θα παράγει χρονοκαθυστέρηση:

t_{DEL}=2.0+555553x1.8+0.8+1.6=2.0+999995.6+0.8+1.6
=999999.8≈1.000.000=1sec

Παράρτημα ΙΙ
Χρόνοι Εκτέλεσης Εντολών

Στο παράρτημα αυτό περιέχονται οι χρόνοι εκτέλεσης των εντολών του 68000 αναφορικά με τις περιόδους του ρολογιού.
Στα δεδομένα που ακολουθούν υποτίθεται ότι οι κύκλοι ανάγνωσης και εγγραφής μνήμης αποτελούνται από τέσσερις περιόδους ρολογιού.
Μακρύτεροι κύκλοι μνήμης θα προκαλέσουν τη γένεση καταστάσεων αναμονής οι οποίες πρέπει να προστεθούν στο συνολικό χρόνο εκτέλεσης εντολής.

Στα δεδομένα περιέχονται μέσα σε παρένθεση (R/W) και οι αριθμοί των κύκλων ανάγνωσης και εγγραφής κάθε εντολής.

Έτσι, για παράδειγμα, η πληροφορία **18(3/1)** δηλώνει ότι ο χρόνος εκτέλεσης της εντολής είναι 18 περίοδοι ρολογιού.

Δηλαδή, 3x4=12 περίοδοι για την ανάγνωση, 1x4=4 για την εγγραφή και δύο περίοδοι για κάποια εσωτερική λειτουργία του μικροεπεξεργαστή.

Χρόνοι εκτέλεσης εντολής MOVE για byte και word

EX346 EX347

Πηγή	Προορισμός								
	Dn	**An**	**(An)**	**(An)+**	**-(An)**	**d(An)**	**d(A0, Ri)**	**XXX.W**	**XXX.L**
Dn	**4(1/0)**	**4(1/0)**	**8(1/1)**	**8(1/1)**	**8(1/1)**	**12(2/1)**	**14(2/1)**	**12(2/1)**	**16(3/1)**
An	**4(1/0)**	**4(1/0)**	**8(1/1)**	**8(1/1)**	**8(1/1)**	**12(2/1)**	**14(2/1)**	**12(2/1)**	**16(3/1)**
(An)	**8(2/0)**	**8(2/0)**	**12(2/1)**	**12(2/1)**	**12(2/1)**	**16(3/1)**	**18(3/1)**	**16(3/1)**	**20(4/1)**
(An)+	**8(2/0)**	**8(2/0)**	**12(2/1)**	**12(2/1)**	**12(2/1)**	**16(3/1)**	**18(3/1)**	**16(3/1)**	**20(4/1)**
-(An)	**10(2/0)**	**10(2/0)**	**14(2/1)**	**14(2/1)**	**14(2/1)**	**18(3/1)**	**20(3/1)**	**18(3/1)**	**22(4/1)**
d(An)	**12(3/0)**	**12(3/0)**	**16(3/1)**	**16(3/1)**	**16(3/1)**	**20(4/1)**	**22(4/1)**	**20(4/1)**	**24(5/1)**
d(A0, Ri)	**14(3/0)**	**14(3/0)**	**18(3/1)**	**18(3/1)**	**18(3/1)**	**22(4/1)**	**24(4/1)**	**22(4/1)**	**26(5/1)**
XXX.W	**12(3/0)**	**12(3/0)**	**16(3/1)**	**16(3/1)**	**16(3/1)**	**20(4/1)**	**22(4/1)**	**20(4/1)**	**24(5/1)**
XXX.L	**16(4/0)**	**16(4/0)**	**20(4/1)**	**20(4/1)**	**20(4/1)**	**24(5/1)**	**26(5/1)**	**24(5/1)**	**28(6/1)**
d(PC)	**12(3/0)**	**12(3/0)**	**16(3/1)**	**16(3/1)**	**16(3/1)**	**20(4/1)**	**22(4/1)**	**20(4/1)**	**24(5/1)**
d(PC, Ri)	**14(3/0)**	**14(3/0)**	**18(3/1)**	**18(3/1)**	**18(3/1)**	**22(4/1)**	**24(4/1)**	**22(4/1)**	**26(5/1)**
#XXX	**8(2/0)**	**8(2/0)**	**12(2/1)**	**12(2/1)**	**12(2/1)**	**16(3/1)**	**18(3/1)**	**16(3/1)**	**20(4/1)**

Χρόνοι εκτέλεσης εντολής MOVE για long word

Πηγή	**Προορισμός**								
	Dn	**An**	**(An)**	**(An)+**	**-(An)**	**d(An)**	**d(A0, Ri)**	**XXX.W**	**XXX.L**
Dn	**4(1/0)**	**4(1/0)**	**12(1/2)**	**12(1/2)**	**12(1/2)**	**16(2/2)**	**18(2/2)**	**16(2/2)**	**20(3/2)**
An	**4(1/0)**	**4(1/0)**	**12(1/2)**	**12(1/2)**	**12(1/2)**	**16(2/2)**	**18(2/2)**	**16(2/2)**	**20(3/2)**
(An)	**12(3/0)**	**12(3/0)**	**20(3/2)**	**20(3/2)**	**20(3/2)**	**24(4/2)**	**26(4/2)**	**24(4/2)**	**28(4/2)**
(An)+	**12(3/0)**	**12(3/0)**	**20(3/2)**	**20(3/2)**	**20(3/2)**	**24(4/2)**	**26(4/2)**	**24(4/2)**	**28(4/2)**
-(An)	**14(3/0)**	**14(3/0)**	**22(3/2)**	**22(3/2)**	**22(3/2)**	**26(4/2)**	**28(4/2)**	**26(4/2)**	**30(5/2)**
d(An)	**16(4/0)**	**16(4/0)**	**24(4/2)**	**24(4/2)**	**24(4/2)**	**28(5/2)**	**30(5/2)**	**28(5/2)**	**32(6/2)**
d(A0, Ri)	**18(4/0)**	**18(4/0)**	**26(4/2)**	**26(4/2)**	**26(4/2)**	**30(5/2)**	**32(5/2)**	**30(5/2)**	**34(6/2)**
XXX.W	**16(4/0)**	**16(4/0)**	**24(4/2)**	**24(4/2)**	**24(4/2)**	**28(5/2)**	**30(5/2)**	**28(5/2)**	**32(6/2)**
XXX.L	**20(5/0)**	**20(5/0)**	**28(5/2)**	**28(5/2)**	**28(5/2)**	**32(6/2)**	**34(6/2)**	**32(6/2)**	**36(7/2)**
d(PC)	**16(4/0)**	**16(4/0)**	**24(4/2)**	**24(4/2)**	**24(4/2)**	**28(5/2)**	**30(5/2)**	**28(5/2)**	**32(5/2)**
d(PC, Ri)	**18(4/0)**	**18(4/0)**	**26(4/2)**	**26(4/2)**	**26(4/2)**	**30(5/2)**	**32(5/2)**	**30(5/2)**	**34(6/2)**
#XXX	**12(3/0)**	**12(3/0)**	**20(3/2)**	**20(3/2)**	**20(3/2)**	**24(4/2)**	**26(4/2)**	**24(4/2)**	**28(5/2)**

Σημειώσεις: *Το μέγεθος του καταχωρητή δείκτη (Ri) δεν επηρεάζει το χρόνο εκτέλεσης της εντολής*

Χρόνοι εκτέλεσης εντολών απευθείας δεδομένων

Εντολή	Μέγεθος	op # , Dn	op # , An	op # , M
ADDI	Byte, Word	**8**(2/0)	-	**12**(2/1)+
	Long	**16**(3/0)	-	**20**(3/2)+
ADDQ	Byte, Word	**4**(1/0)	**8**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+
ANDI	Byte, Word	**8**(2/0)	-	**12**(2/1)+
	Long	**16**(3/0)	-	**20**(3/1)+
CMPI	Byte, Word	**8**(2/0)	-	**8**(2/0)+
	Long	**14**(3/0)+	-	**12**(2/1)+
EORI	Byte, Word	**8**(2/0)	-	**8**(1/1)+
	Long	**16**(3/0)	-	**20**(3/2)+
MOVEQ	Long	**4**(1/0)	-	-
ORI	Byte, Word	**8**(2/0)	-	**12**(2/1)+
	Long	**16**(3/0)	-	**20**(3/2)+
SUBI	Byte, Word	**8**(2/0)	-	**12**(2/1)+
	Long	**16**(3/0)	-	**20**(3/2)+
SUBQ	Byte, Word	**4**(1/0)	**8**(1/0)*	**8**(1/1)+
	Long	**8**(1/0)	**8**(1/0)	**12**(1/2)+

Σημειώσεις:

+ Πρόσθεσε το χρόνο υπολογισμού της ενεργού διεύθυνσης.

Χρόνοι εκτέλεσης εντολών υπό συνθήκη

Εντολή	Μετατόπιση	Γίνεται διακλάδωση	Δε γίνεται διακλάδωση
B_{CC}	Byte	**10**(2/0)	**8**(1/0)
	Word	**10**(2/0)	**12**(2/0)
BRA	Byte	**10**(2/0)	**-**
	Word	**10**(2/0)	**-**
BSR	Byte	**18**(2/2)	**-**
	Word	**18**(2/2)	**-**
DB_{CC}	CC αληθινό	**-**	**12**(2/0)
	CC όχι αληθινό	**18**(2/2)	**14**(3/0)

Χρόνοι εκτέλεσης συμπληρωματικών εντολών

Εντολή	Μετατόπιση	Καταχωρητής	Μνήμη
ANDI to CCR	Byte	**20**(3/0)	-
ANDI to SR	Word	**20**(3/0)	-
CHK	-	**10**(1/0)+	-
EORI to CCR	Byte	**20**(3/0)	-
EORI to SR	Word	**20**(3/0)	-
ORI to CCR	Byte	**20**(3/0)	-
ORI to SR	Word	**20**(3/0)	-
MOVE from SR	-	**6**(1/0)	**8**(1/1)+
MOVE to CCR	-	**12**(2/0)	**12**(2/0)+
MOVE to SR	-	**12**(2/0)	**12**(2/0)+
EXG	-	**6**(1/0)	-
EXT	Word	**4**(1/0)	-
	Long	**4**(1/0)	-
LINK	-	**16**(2/2)	-
MOVE from USP	-	**4**(1/0)	-
MOVE to USP		**4**(1/0)	-
NOP	-	**4**(1/0)	-
RESET	-	**132**(1/0)	-
RTE	-	**20**(5/0)	-
RTR	-	**20**(5/0)	-
RTS	-	**16**(4/0)	-
STOP	-	**4**(0/0)	-
SWAP		**4**(1/0)	-
TRAPV		**4**(1/0)	-
UNLK	-	**12**(3/0)	-

Σημειώσεις:

+ Πρόσθεσε το χρόνο υπολογισμού της ενεργού διεύθυνσης.

Θέματα Θεωρίας Κεφαλαίου ΙΙΙ

Θέμα 1ο

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ     #$5C,D0
CMPI.B    #$60,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=1 C=1 Z=0 [D0]=0000005C

```
MOVEQ     #$5C,D0
SUBI.B    #$60,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=1 C=1 Z=0 [D0]=000000FC

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ     #$2E,D0
CMPI.B    #$25,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=0, C=0, Z=0 [D0]=$0000002E

```
MOVEQ     #$2E,D0
SUBI.B    #$25,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=0, C=0, Z=0, [D0]=00000009

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ    #$7C,D0
CMPI.B   #$32,D0
BEQ      LABEL
MOVE.B   D0,$400400
.
.
.
```

N=0 C=0 Z=0 [D0]=$0000007C

```
MOVEQ    #$7C,D0
SUBI.B   #$32,D0
BEQ      LABEL
MOVE.B   D0,$400400
.
.
.
```

N=0 C=0 Z=0 [D0]=$0000004A

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ     #$5C,D0
CMPI.B    #$60,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=1 C=1 Z=0   [D0]=0000005C

```
MOVEQ     #$5C,D0
SUBI.B    #$60,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
.
.
```

N=1 C=1 Z=0   [D0]=000000FC

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ     #$B9,D0
CMP.B     #$B9,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
```

N=0 C=0 Z=1 [D0]=$FFFFFFB9
[$400400]=$B9

```
MOVEQ     #$B9,D0
SUBI.B    #$B9,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
```

N=0 C=0 Z=1 [D0]=$FFFFFF00
[$400400]=$00

Παρακάτω φαίνονται δύο κομμάτια προγράμματος που κάνουν την ίδια εργασία. Ποια θα είναι τα περιεχόμενα του καταχωρητή D0 μετά την εκτέλεσή τους.

```
MOVEQ     #$A8,D0
CMP.B     #$AA,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
```

N=1 C=1 Z=0 [D0]=$FFFFFFA8
[$400400]=$A8

```
MOVEQ     #$A8,D0
SUBI.B    #$AA,D0
BEQ       LABEL
MOVE.B    D0,$400400
.
```

N=1 C=1 Z=0 [D0]=$FFFFFFFE
[$400400]=$FE

Θέμα 2ο

Υποθέστε ότι [D0]=4 και [D1]=HHHHHHA9 και οι δείκτες του καταχωρητή κατάστασης έχουν τις τιμές X=1 και C=1. Να υπολογισθούν τα περιεχόμενα των D1, C και X:

A. Μετά από την εκτέλεση της εντολής: **LSL.B D0,D1**
[D1]=HHHHHH90 C=0 X=0

B. Μετά από την εκτέλεση της εντολής: **ROXL.B D0,D1**
[D1]=HHHHHH9D C=0 X=0

Υποθέστε ότι [D0]=3 και [D1]=HHHHHHB5 και οι δείκτες του καταχωρητή κατάστασης έχουν τις τιμές X=0 και C=1. Να υπολογισθούν τα περιεχόμενα των D1, C και X:

A. Μετά από την εκτέλεση της εντολής: **ASR.B D0,D1**
[D1]=$HHHHHHF6 C=1 X=1
B. Μετά από την εκτέλεση της εντολής: **ROXL.B D0,D1**
[D1]=$HHHHHHAA C=1 X=1

Υποθέστε ότι [D0]=4 και [D1]=HHHHHHA1 και οι δείκτες του καταχωρητή κατάστασης έχουν τις τιμές X=0 και C=1. Να υπολογισθούν τα περιεχόμενα των D1, C και X:

A. Μετά από την εκτέλεση της εντολής: **ASL.B D0,D1**
[D1]=$HHHHHH10, C=0, X=0
B. Μετά από την εκτέλεση της εντολής: **ROXR.B D0,D1**
[D1]=$HHHHHH2A, C=0, X=0

Υποθέστε ότι [D0]=2 και [D1]=HHHH237D και οι δείκτες του καταχωρητή κατάστασης έχουν τις τιμές X=0 και C=1. Να υπολογισθούν τα περιεχόμενα των D1, C και X:

A. Μετά από την εκτέλεση της εντολής: **LSL.B D0,D1** ***[D1]=$HHHH23F4 C=1 X=1***
B. Μετά από την εκτέλεση της εντολής: **LSR.W D0,D1** ***[D1]=$HHHH08DF C=0 X=0***
C. Μετά από την εκτέλεση της εντολής: **ROXL.W D0,D1*****[D1]=$HHHH8DF6 C=0 X=0***
D. Μετά από την εκτέλεση της εντολής: **ROXR.B D0,D1*****[D1]=$HHHH23F6 C=1 X=1***

Υποθέστε ότι [D0]=3 και [D1]=HHHHA39A και οι δείκτες του καταχωρητή κατάστασης έχουν τις τιμές X=1 και C=1. Να υπολογισθούν τα περιεχόμενα των D1, C και X:

A. Μετά από την εκτέλεση της εντολής: **LSL.B D0,D1** ***[D1]=$HHHH A3D0 C=0 X=0***
B. Μετά από την εκτέλεση της εντολής: **LSR.W D0,D1** ***[D1]=$HHHH1473 C=0 X=0***
C. Μετά από την εκτέλεση της εντολής: **ROXL.W D0,D1*****[D1]=$HHHH1CD6 C=1 X=1***
D. Μετά από την εκτέλεση της εντολής: **ROXR.B D0,D1*****[D1]=$HHHHA3B3 C=0 X=0***

Θέμα 3ο

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $4E. Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

MOVE.B $400400,D0
ANDI.B #$28,D0
[D0]=$HHHHHH08, [N]=0, [Z]=0

Αλλάξτε την **ANDI** με **ORI** και επαναλάβατε το προηγούμενο.
[D0]=$HHHHHH6E, [N]=0, [Z]=0

Αλλάξτε την **ORI** με **EORI** και επαναλάβατε το ίδιο.
[D0]=$HHHHHH66, [N]=0, [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $AB.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

MOVE.B $400400,D0

ANDI.B #$28,D0

[D0]=HHHHHH28 [N]=0 [Z]=0

Αλλάξτε την **ANDI** με **ORI** και επαναλάβατε το προηγούμενο.

[D0]=HHHHHHAB [N]=1 [Z]=0

Αλλάξτε την **ORI** με **EORI** και επαναλάβατε το ίδιο.

[D0]=HHHHHH83 [N]=1 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $6B.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

α. MOVE.B $400400,D0

ANDI.B #$90,D0

[D0]=$HHHHHH00 [N]=0 [Z]=1

β. MOVE.B $400400,D0

ORI.B #$71,D0

[D0]=$HHHHHH7B [N]=0 [Z]=0

γ. MOVE.B $400400,D0

EORI.B #$EB,D0

[D0]=$HHHHHH80 [N]=1 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $5F.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

MOVE.B $400400,D0
ANDI.B #$A9,D0

[D0]=HHHHHH09 [N]=0 [Z]=0

Αλλάξτε την **ANDI** με **ORI** και επαναλάβατε το προηγούμενο.

[D0]=HHHHHHFF [N]=1 [Z]=0

Αλλάξτε την **ORI** με **EORI** και επαναλάβατε το ίδιο.

[D0]=HHHHHHF6 [N]=1 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $7A.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

α. **MOVE.B $400400,D0**

ANDI.B #$84,D0

[D0]=$HHHHHH00 [N]=0 [Z]=1

β. **MOVE.B $400400,D0**

ORI.B #$C6,D0

[D0]=$HHHHHHFE [N]=1 [Z]=0

γ. **MOVE.B $400400,D0**

EORI.B #$43,D0

[D0]=$HHHHHH39 [N]=0 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $5A.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

MOVE.B $400400,D0
ANDI.B #$A4,D0

[D0]=$HHHHHH00 [N]=0 [Z]=1

Αλλάξτε την **ANDI** με **ORI** και επαναλάβατε το προηγούμενο.

[D0]=$HHHHHHFE [N]=1 [Z]=0

Αλλάξτε την **ORI** με **EORI** και επαναλάβατε το ίδιο.

[D0]=$HHHHHHFE [N]=1 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $AB.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

MOVE.B $400400,D0

ANDI.B #$28,D0

[D0]=HHHHHH28 [N]=0 [Z]=0

Αλλάξτε την **ANDI** με **ORI** και επαναλάβατε το προηγούμενο.

[D0]=HHHHHHAB [N]=1 [Z]=0

Αλλάξτε την **ORI** με **EORI** και επαναλάβατε το ίδιο.

[D0]=HHHHHH83 [N]=1 [Z]=0

Υποτίθεται ότι στη θέση μνήμης $400400 είναι αποθηκευμένη η πληροφορία $6B.Υπολογίστε τα περιεχόμενα του D0 και τις τιμές των N και Z μετά την εκτέλεση της παρακάτω ακολουθίας εντολών.

α. **MOVE.B $400400,D0**

ANDI.B #$90,D0

[D0]=$HHHHHH00 [N]=0 [Z]=1

β. **MOVE.B $400400,D0**

ORI.B #$71,D0

[D0]=$HHHHHH7B [N]=0 [Z]=0

γ. **MOVE.B $400400,D0**

EORI.B #$EB,D0

[D0]=$HHHHHH80 [N]=1 [Z]=**0**

Θέμα 4ο

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$A7,D0**	***[D0]=$FFFFFFA7***
MOVE.B	**#$A7,D0**	***[D0]=$HHHHHHA7***
MOVE.B	**$400418,D0**	***[D0]=$HHHHHHB0***
MOVE.W	**$400418,D0**	***[D0]=$HHHHB0BF***
MOVE.L	**$400418,D0**	***[D0]=$B0BFC9DD***

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$45,D0**	***[D0]=$00000045***
MOVE.B	**$400406,D0**	***[D0]=$HHHHHH00***
MOVE.W	**$400406,D0**	***[D0]=$HHHH005F***
MOVE.L	**$400406,D0**	***[D0]=$005FB0BF***

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$A7,D0**	**[D0]= *$FFFFFFA7***
MOVE.B	**$40041A,D0**	**[D0]= *$HHHHHHC9***
MOVE.W	**$40041A,D0**	**[D0]= *$HHHHC9DD***
MOVE.L	**$40041A,D0**	**[D0]= *$C9DDECE0***

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$3E,D0**	***[D0]=$0000003E***
MOVE.B	**$400408,D0**	***[D0]=$HHHHHHB0***
MOVE.W	**$400408,D0**	***[D0]=$HHHHB0BF***
MOVE.L	**$400408,D0**	***[D0]=$B0BFC9DD***

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$B0,D0**	***[D0]=$FFFFFFB0***
MOVE.B	**$400418,D0**	***[D0]=$HHHHHHB0***
MOVE.W	**$400418,D0**	***[D0]=$HHHHB0BF***
MOVE.L	**$400418,D0**	***[D0]=$B0BFC9DD***

Αν υποτεθεί ο παρακάτω πίνακας μνήμης,

400400	400401	400402	400403	400404	400405	400406	400407
00	19	35	47	5D	57	00	5F
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
B0	BF	C9	DD	EC	E0	D2	C4
400410	400411	400412	400413	400414	400415	400416	400417
00	19	35	47	5D	57	00	5F
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F
B0	BF	C9	DD	EC	E0	D2	C4

Ποια θα είναι τα περιεχόμενα του D0 μετά την εκτέλεση κάθε μιας από τις παρακάτω εντολές.

MOVEQ	**#$A5,D0**	***[D0]=FFFFFFA5***
MOVE.B	**#$A5,D0**	***[D0]=HHHHHHA5***
MOVE.B	**$40041C,D0**	***[D0]=HHHHHHEC***
MOVE.W	**$40041C,D0**	***[D0]=HHHHECE0***
MOVE.L	**$40041C,D0**	***[D0]=ECE0D2C4***

Θέμα 5ο

Να συμπληρωθεί ο πίνακας μνήμης μετά τη συμβολομετάφραση του κώδικα:

```
        ORG   $400412
        DC.W   $45,$3E
        DS.B 2
        ORG $40041A
        MOVE.B $400500,A0
LOOP:   MOVE.B (A0)+,D0
        BNE LOOP
        RTS
```

Μνήμη		
Διεύθυνση	Περιεχόμενο	
400410		
400411		
400412	00	
400413	$45	
400414	00	
400415	$3E	
400416	Κενό	
400417	Κενό	
400418		
400419		
40041A		Κωδικός εντολής MOVE.B $400500
40041B		
40041C	00	
40041D	40	
40041E	05	
40041F	00	
400420		Κωδικός εντολής MOVE.B (A0)+,D0
400421		
400422		Κωδικός εντολής BNE LOOP
400423		
400424		Κωδικός εντολής RTS
400425		
400426		
400427		
400428		

Να συμπληρωθεί ο πίνακας μνήμης μετά τη συμβολομετάφραση του παρακάτω κώδικα:

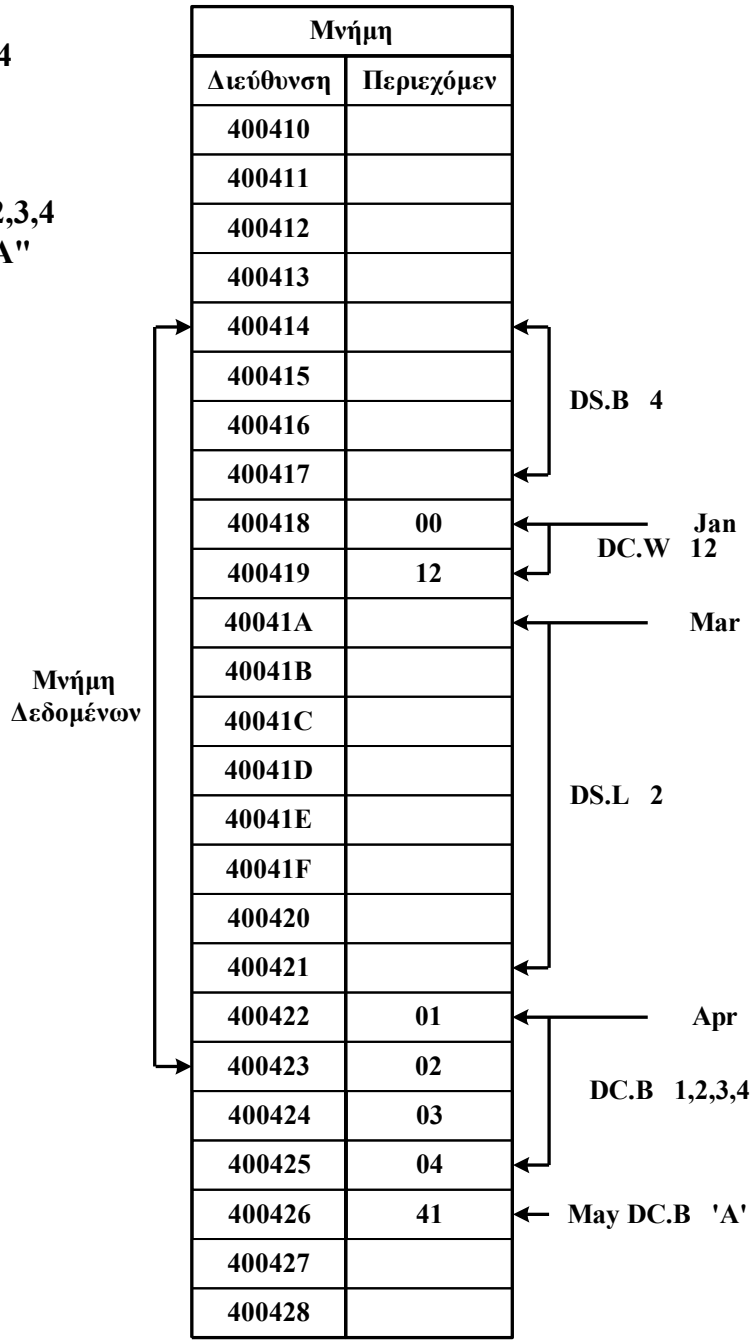
```
                ORG         $400410
NUMS            DC.B        $28,$A4
RESULT          DS.W        1
SUM             MOVE.B      D0,NUMS

                ADD.B       D0,NUMS+1
                MOVE.B      D0, RESULT
                .
                .
                .
```

Μνήμη		
Διεύθυνση	Περιεχόμενο	
400410	$28	DC.B $28,$A4
400411	$A4	
400412		DS.W 1
400413		
400414		Κωδικός εντολής MOVE.B D0,NUMS
400415		
400416	00	Διεύθυνση NUMS
400417	40	
400418	04	
400419	10	
40041A		Κωδικός εντολής MOVE.B D0,NUMS+1
40041B		
40041C	00	Διεύθυνση NUMS+1
40041D	40	
40041E	04	
40041F	11	
400420		Κωδικός εντολής MOVE.B D0,RESULT
400421		
400422	00	Διεύθυνση RESULT
400423	40	
400424	00	
400425	12	
400426		
400427		
400428		

Να συμπληρωθεί ο πίνακας μνήμης μετά τη συμβολομετάφραση του παρακάτω κώδικα:

```
ORG   $400414
DS.B   4
Jan DC.W   12
Mar DS.L   2
Apr DC.B   1,2,3,4
May DC.B   "A"
```



Να συμπληρωθεί ο πίνακας μνήμης μετά τη συμβολομετάφραση του παρακάτω κώδικα:

```
                ORG $400410
NUM1            DC.B $24
NUM2            DC.B $A7
SUM             DS.W 1
                ORG $400414
                MOVE.B   NUM1,D0
                ADD.B    NUM2,D0
                MOVE.B   D0,SUM
                RTS
```

Μνήμη	
Διεύθυνση	Περριεχόμενο
400410	24
400411	A7
400412	CB
400413	
400414	
400415	
400416	00
400417	40
400418	04
400419	10
40041A	
40041B	
40041C	00
40041D	40
40041E	04
40041F	11
400420	
400421	
400422	00
400423	40
400424	04
400425	12
400426	
400427	
400428	

Κωδικός εντολής MOVE.B

Κωδικός εντολής ADD.B

Κωδικός εντολής MOVE.B

Να συμπληρωθεί ο πίνακας μνήμης μετά τη συμβολομετάφραση του παρακάτω κώδικα:

```
        ORG       $400400
NUMS    DC.W      $3EE2,$1111
RESULT  DS.L      1
        ORG       $400408
SUBNS   MOVE.W    NUMS,D0
        MOVE.W    NUMS+1,D1
        SUB.W     D0,D1
        MOVE.W    D1,RESULT
        RTS
```

Μνήμη	
Διεύθυνση	**Περιεχόμ.**
400400	3E
400401	E2
400402	11
400403	11
400404	2D
400405	D1
400406	
400407	
400408	00
400409	40
40040A	04
40040B	00
40040C	
40040D	
40040E	00
40040F	40
400410	04
400411	02
400412	
400413	
400414	
400415	
400416	00
400417	40
400418	04
400419	04

Κωδικός εντολής MOVE.W NUMS,D0

Κωδικός εντολής MOVE.W NUMS+1,D1

Κωδικός εντολής SUB.W D0,D1

Κωδικός εντολής MOVE.W D1,RESULT

```
        ORG    $400410
NUMS    DC.B   $28,$A4
RESULT  DS.W   1
SUM     MOVE.B D0,NUMS
        ADD.B  D0,NUMS+1
        MOVE.B D0, RESULT
        END
```

Να συμπληρωθεί ο πίνακας μνήμης μετά την εκτέλεση του κώδικα:

Μνήμη	
Διεύθυνση	Περιεχόμενο
400410	28
400411	A4
400412	CC
400413	00
400414	
400415	
400416	00
400417	40
400418	04
400419	10
40041A	
40041B	
40041C	00
40041D	40
40041E	04
40041F	11
400420	
400421	
400422	00
400423	40
400424	04
400425	12
400426	
400427	
400428	

Κωδικός εντολής MOVE.B D0,NUMS

Κωδικός εντολής MOVE.B D0,NUMS+1

Κωδικός εντολής MOVE.B D0,RESULT

Θέμα 6ο

Αναφέρατε τις μεθόδους διευθυνσιοδότησης για τους τελεστέους προέλευσης και προορισμού των παρακάτω εντολών.

		Προέλευσης	Προορισμού
1. MOVE.W	**D3,D2**	***Άμεση Κατ. Δεδομένων***	***Άμεση Κατ. Δεδομένων***
2. MOVE.B	**D3,A2**	***Άμεση Κατ. Δεδομένων***	***Άμεση Κατ. Διευθ.***
3. MOVE.L	**D3,$400400**	***Άμεση Κατ. Δεδομένων***	***Απόλυτη μακρ. Δεδομ.***
4. MOVE.L	**#100,D2**	***Απευθείας Δεδομένων***	***Άμεση Κατ. Δεδομένων***
5. MOVE.W	**#$A10D,$1122**	***Απευθείας Δεδομένων***	***Απόλυτη κοντών δεδομ.***
6. MOVE.B	**D3,(A2)**	***Άμεση Κατ. Δεδομένων***	***Έμμεση καταχωρητή***
7. MOVE.L	**A1,(A2)+**	***Άμεση Κατ. Διευθύνσεων***	***Έμμεση μεταυξητική***
8. MOVE.L	**-(A2),D3**	***Έμμεση προμειωτική***	***Άμεση Κατ. Δεδομένων***
9. MOVE.W	**10(A2),D3**	***Έμμεση με μετατόπιση***	***Άμεση Κατ. Δεδομένων***

Αναφέρατε τις μεθόδους διευθυνσιοδότησης για τους τελεστέους προέλευσης και προορισμού των παρακάτω εντολών.

		Προέλευσης	Προορισμού
1.	MOVE.W #$3F42,D2	*Απευθείας Δεδομένων*	*Άμεση Κατ. Δεδομένων*
2.	MOVE.L D3,A2	*Άμεση Κατ. Δεδομένων*	*Άμεση Κατ. Διευθύνσεων*
3.	MOVE.L $400400,D0	*Απόλυτη μακρ. δεδομένων*	*Άμεση Κατ. Δεδομένων*
4.	MOVE.B $10(A0),D2	*Έμμεση με μετατόπιση*	*Άμεση Κατ. Δεδομένων*
5.	MOVE.W #1,$112200	*Απευθείας Δεδομένων*	*Απόλυτη μακρ. δεδομένων*
6.	MOVE.B D3,(A2)	*Άμεση Κατ. Δεδομένων*	*Έμμεση καταχωρητή*
7.	MOVE.L A1,(A2)+	*Άμεση Κατ. Διευθύνσεων*	*Έμμεση μεταυξητική*
8.	MOVE.L -(A2),D3	*Έμμεση προμειωτική*	*Άμεση Κατ. Δεδομένων*
9.	MOVE.W 10(A2,D1),D3	*Έμμ. με δείκτη και μετατ.*	*Άμεση Κατ. Δεδομένων*

Αναφέρατε τις συνθήκες που κάνουν τις παρακάτω εντολές ισοδύναμες.

α. MOVE.W	**D0,ABCD**	***D0=ABCD***
β. MOVE.W	**D0,$10(A1)**	***D0=ABCD αν [A1+10]=ABCD***
γ. MOVE.W	**D0,$100(A2,D1.L)**	***D0=ABCD αν [A2+D1+100]=ABCD***
δ. MOVE.W	**D0,(A3)**	***D0=ABCD αν [A3]=ABCD***

Ποιες από τις παρακάτω εντολές είναι σωστές και ποιες λανθασμένες;

		Σωστή	Λανθασμένη
1. ANDI	**D4,D5**		***Λανθασμένη***
2. ADDQ	**#8,A3**	***Σωστή***	
3. MOVEA.B	**#4,A3**		***Λανθασμένη***
4. MOVEQ	**#50,D5**	***Σωστή***	
5. DS.B	**1,2**		***Λανθασμένη***
6. ORG	**#400400**		***Λανθασμένη***
7. BEQ.B	**LOOP**		***Λανθασμένη***
8. SUBI.B	**1,D1**		***Λανθασμένη***
9. DC.W	**3,4**	***Σωστή***	

Σημειώστε ποιες από τις παρακάτω εντολές είναι σωστές και διορθώστε τις λανθασμένες.

		Σωστή	Διορθωμένη
1. MOVE.B	**LABEL,#$4**	_____	***MOVE.B #$4, LABEL***
2. ADD.B	**#1,A3**	_____	***ADDA.W #1,A3***
3. CMPI.B	**D0,#9**	_____	***CMPI.B #9,D0***
4. MOVEQ	**#50,D5**	***Σωστή***	
5. DS.B	**$1,$2**	_____	***DS.B $1 ή DS.B $2***
6. ORG	**#400400**	_____	***ORG $400400***
7. BEQ.B	**LOOP**	_____	***BEQ LOOP***
8. ADDQ.B	**1,D1**	_____	***ADDQ #1,D1***
9. DC.W	**$3,$4**	***Σωστή***	

Οι παρακάτω εντολές είναι όλες λαθεμένες. Ποιο είναι το λάθος σε κάθε περίπτωση;

α. ADDQ.L	#12,D2	***ADDQ.L #(από1-8),D2***
β. ANDI.B	#FC,D6	***ANDI.B #$FC,D6***
γ. ASL.L	#9,D3	***ASL.L #(από 1-8),D3***
δ. NOT.W	D3,D7	***NOT.W D3 ή NOT.W D7***

Οι παρακάτω εντολές είναι όλες λαθεμένες. Γράψτε τις διορθωμένες.

α. MOVE	**Temp,#$400400**	***MOVEA.L Temp,$400400***
β. ADD.B	**#1,A3**	***ADDI.B #1,D3 (όχι κατ. Διεύθυνσης)***
γ. CMP.L	**D0,#9**	***CMPI.L #9,D0***
δ. MOVE.B	**#500,D5**	***MOVE.W #500,D5***
ε. DS.B	**1,2**	***DS.B 1 ή DS.B 2***
στ. ORG	**#400400**	***ORG $400400***
ζ. BEQ.B	**LOOP**	***BEQ LOOP***
η. ADDQ	**1,D1**	***ADDQ.B #1,D1***
θ. DC.W	**3,4**	***DC.B $3,$4***
i. MOVE.W	**$400401,D0**	***MOVE.W $400400,D0 ή MOVE.B $400401,D0***

Οι παρακάτω εντολές είναι όλες λαθεμένες. Ποιο είναι το λάθος σε κάθε περίπτωση;

1. MULU	**#$*12345*,D0**	***MULU #$1234,D0***
2. ADDQ	**#*9*,A3**	***ADDQ #7,A3***
3. CMPI.L	**$40041F,D3**	***CMPI.L #$40041F,D3***
4. MOVEQ	**#$*4550*,D5**	***MOVEQ #$(byte),D5***
5. DS.B	**1,2**	***DS.B 1 ή DS.B 2***
6. ORG	**#400**	***ORG $400***
7. BEQ.*B*	**LOOP**	***BEQ LOOP***
8. ADDI.B	**#*256*,D1**	***ADDI.B #255 (ή μικρότερο) ,D1***

Οι παρακάτω εντολές είναι όλες λαθεμένες. Ποιο είναι το λάθος σε κάθε περίπτωση;

α. ADDQ.L #12,D2 ***από 1 έως 8***

β. ANDI.B #FC,D6 ***ANDI.B #$FC,D6***

γ. ASL.L #9,D3 ***από 1 έως 8***

δ. NOT.W D3,D7 ***NOT.W D3 ή NOT.W D7***

Κάθε μια απ' τις παρακάτω εντολές είναι συμβατές με τον Assembler του 68000. Υποθέτοντας ότι όλες οι διευθύνσεις αναφέρονται σε byte, που εκφράζονται στο δεκαδικό σύστημα, και ότι [D0]=0, [A0]=4, [A1]=2, και [PC]=10. Χρησιμοποιώντας τον παρακάτω πίνακα μνήμης, εξηγήστε τη λειτουργία των παρακάτω εντολών.

Σημείωση: *Η πρώτη σειρά του πίνακα είναι οι θέσεις μνήμης και δεύτερη το αντίστοιχο περιεχόμενο.*

1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	7	2	5	2	0	7	9	5	4	2	4	6	1

Εντολές:		***Απαντήσεις:***
α. LEA	**(A0),A3**	***A3=5***
b. LEA	**(-2,A0),A3**	***A3=7***
c. MOVE.B	**(4,PC),D2**	***D2=1***
δ. MOVE.B	**(2,PC,A1),D7**	***D7=1***
ε. LEA	**(10,A0,D0),A3**	***A3=1***
f. ADD.B	**12,D0**	***D0=4***
g. MOVE.B	**(-1,A0,A1),D4**	***D4=2***
h. ADD.B	**D0,(4,A0,A1)**	***Θέση μνήμης 10=4***
j. MOVE.B	**8,D4**	***D4=9***
k. MOVE.B	**#3,(A0)+**	***Θέση μνήμης 4=3, A0=5***

Θέμα 7ο

Δώστε την δεκαδική τιμή του δυαδικού αριθμού 10101001_2 όταν,

α. Ερμηνεύεται σαν ένας μη προσημασμένος αριθμός. Τιμή=***169***
β. Ερμηνεύεται σαν ένας προσημασμένος αριθμός. Τιμή=***-87***

Δώστε την δεκαδική τιμή του δυαδικού αριθμού 10110101_2 όταν,

α. Ερμηνεύεται σαν ένας μη προσημασμένος αριθμός. ***Τιμή= 181***
β. Ερμηνεύεται σαν ένας προσημασμένος αριθμός. ***Τιμή= -75***

Δώστε την δεκαδική τιμή του δυαδικού αριθμού 11011101_2 όταν,

α. Ερμηνεύεται σαν ένας μη προσημασμένος αριθμός. ***Τιμή= 221***
β. Ερμηνεύεται σαν ένας προσημασμένος αριθμός. ***Τιμή= -35***

Δώστε την δεκαδική τιμή του δυαδικού αριθμού 10100001_2 όταν,

α. Ερμηνεύεται σαν ένας μη προσημασμένος αριθμός. ***Τιμή= 161***

β. Ερμηνεύεται σαν ένας προσημασμένος αριθμός. ***Τιμή= -95***

Θέμα 8ο

Ποια είναι η διαφορά μεταξύ “εκτελεστέας εντολής” και “ψευδεντολής”;

Η ψευδεντολή σε αντίθεση με την εκτελεστέα εντολή δεν είναι εντολή που συμπεριλαμβάνεται στη λίστα των εντολών του μικροεπεξεργαστή και επομένως δεν μπορεί να εκτελεστεί απ’ αυτόν. Χρησιμοποιείται για να εντέλει τον συμβολομεταφραστή να κάνει κάποια εργασία κατά τη διάρκεια της συμβολομετάφρασης.

Θέμα 9ο

Δοθέντος ότι [D0]= 87654321, [D1]= 12345678 και [A0]=$00400522, ζητείται να ορίσετε τα περιεχόμενα των θέσεων μνήμης $400400-$400402 μετά την εκτέλεση της εντολής MOVEM.W D0/D1/A0,$400400.

[$400400]= ***43***
[$400401]= ***21***
[$400402]= ***56***
[$400403]= ***78***

Δοθέντος ότι [D0]= 87654321, [D1]= 12345678 και [A0]=$00400522, ζητείται να ορίσετε τα περιεχόμενα των θέσεων μνήμης του πίνακα, που επηρεάζονται, μετά την εκτέλεση της εντολής MOVEM.W D0/D1/A0,$400400.

400400	400401	400402	400403	400404	400405	400406	400407
43	***21***	***56***	***78***	***05***	***22***		
400408	400409	40040A	40040B	40040C	40040D	40040E	40040F
400410	400411	400412	400413	400414	400415	400416	400417
400418	400419	40041A	40041B	40041C	40041D	40041E	40041F

Θέμα 10

Ποια είναι η διαφορά, αν υπάρχει, μεταξύ των παρακάτω δύο προγραμμάτων;

```
A.          ORG     $400400          B.      ORG    $400400
      TEMP  DS.W    1                        DC.W   $6A
            MOVE.W  #$6A,D0
            MOVE.W  D0,TEMP
```

Απάντηση: ***A. Καμία διαφορά.***

B. Η διαφορά έγκειται στο ότι_____

Ποια είναι η διαφορά, αν υπάρχει, μεταξύ των παρακάτω δύο προγραμμάτων;

```
A.          ORG     $400400          B.  ORG    $400400
   TEMP     DS.W    1                    DC.W   $5B
            MOVE.W  #$5B,D0
            MOVE.W  D0,TEMP
```

Απάντηση: ***A. Καμία διαφορά.***

B. Η διαφορά έγκειται στο ότι_____

Θέματα Προγραμμάτων Κεφαλαίου ΙΙΙ

Πρόγραμμα 1ο

Στα δημοτολόγια ενός χωριού είναι καταχωρημένοι 1162 κάτοικοι (inhabitants) στις διευθύνσεις από $400500 και πάνω. Κάθε κάτοικος έχει ένα κωδικό 16 ψηφίων στον οποίο έχουν κωδικοποιηθεί όλα τα στοιχεία της ταυτότητάς του. Το φύλο του κατοίκου κωδικοποιήθηκε με το ψηφίο d15, που είναι 0 για τους άνδρες και 1 για τις γυναίκες.

Να γραφτεί μια υπορουτίνα που θα δημιουργεί δύο λίστες μία για τους άνδρες (males) και μια για τις γυναίκες (females) και θα τις αποθηκεύει την πρώτη απ' τη διεύθυνση $401000 και τη δεύτερη απ' τη διεύθυνση $401800 και πάνω.

```
SBRTN:    MOVE.W    #1162,D0
          LEA       $400500,A0
          LEA       $401000,A1
          LEA       $401800,A2
NEXT:     MOVE.W    (A0)+,D1
          BTST      #15,D1
          BNE       MALE
FEMALE:   MOVE.W    D1,(A2)+
          BRA.S     COUNT
MALE:     MOVE.W    D1,(A1)+
COUNT:    SUBQ      #1,D0
          BNE       NEXT
          RTS
```

Πρόγραμμα 2ο

Να γραφτεί μια υπορουτίνα που θα κάνει τις παρακάτω λειτουργίες: Θα ελέγχει το ένα μετά το άλλο τα 3 λιγότερο σημαντικά ψηφία του καταχωρητή D0 και ανάλογα με τις τιμές τους θα εκτελεί μια απ' τις τρεις σειρές εντολών: SUBA, SUBB, SUBC. Οι σειρές εντολών επιλέγονται με την προτεραιότητα που ακολουθεί:

3 λιγότερο σημαντικά ψηφία	*εκτέλεση*
XX1	*SUBA*
X10	*SUBB*
100	*SUBC*

Αφού η υπορουτίνα εκτελεστεί, και πριν απ' την επιστροφή στο κυρίως πρόγραμμα, το αντίστοιχο ψηφίο του καταχωρητή D0 καθαρίζεται.

```
SUBRTN  MOVE.B  D0,D1
        ANDI.B  #01,D0
        BNE     SUBA
        MOVE.B  D1,D0
        ANDI.B  #2,D0
        BNE     SUBB
        MOVE.B  D1,D0
        ANDI.B  #4,D0
        BNE     SUBC
        BRA.S   SUBRTN


SUBA    .
        ANDI.B  #FE,D1
        MOVE.B  D1,D0
        RTS
SUBB    .
        ANDI.B  #FC,D1
        MOVE.B  D1,D0
        RTS
SUBC    .
        ANDI.B  #F8,D1
        MOVE.B  D1,D0
        RTS
```

Δεύτερος τρόπος καλύτερος

```
SBRTN:  BTST.B   #0,D0
        BNE      SUBA
        BTST.B   #1,D0
        BNE      SUBB
        BTST.B   #2,D0
        BNE      SUBC
        BRA.S    SBRTN

SUBA:   .
        .
        .
        EORI.B   #1,D0
        RTS

SUBB:   .
        .
        .
        EORI.B   #2,D0
        RTS

SUBC:   .
        .
        .
        EORI.B   #4,D0
        RTS
```

Πρόγραμμα 3ο

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     $400400
SUBRTN      MOVE.B  #$0F,D1
            LEA     TABLE1,A0
            LEA     TABLE2,A1
LOOP        MOVE.B  (A0)+,(A1)+
            DBRA    D1, LOOP
            RTS
```

Πρόγραμμα 4ο

Ο Δήμαρχος μια πόλης με εγγεγραμμένους 69.861 πολίτες, διαθέτει στο ταμείο κοινωνικής πολιτικής 165.000 Ευρώ. Τα χρήματα αυτά πρόκειται να τα μοιράσει εξίσου στους πολίτες που είναι άνω των 65 ετών. Οι ηλικίες των πολιτών χρειάζονται ένα byte η κάθε μια και είναι καταχωρημένες με τυχαία σειρά σ' έναν ορμαθό που αρχίζει απ' τη θέση μνήμης $400500. Να γραφτεί ένα πρόγραμμα που θα καταχωρεί στη θέση μνήμης $400400 το ποσό που θα πάρει ο κάθε δικαιούχος σε ακέραια Ευρώ.

```
                ORG         $400400
                AMOUNT:     DS.W    1
                MONEY:      DC.L    165000
                COUNT :     DC.L    69861

                ORG         $400410
SUBROUTINE:     CLR.W       D0
                CLR.L       D1
                MOVE.L      $400500,A0
LOOP:           MOVE.B      (A0)+,D2
                CMPI.B      #65,D2
                BCS         YOUNGER
                ADDQ.W      #1,D0
YOUNGER:        ADDQ.L      #1,D1
                CMP.L       D1,COUNT
                BNE         LOOP
                MOVE.L      MONEY,D3
                DIVU        D0,D3
                MOVE.W      D3,AMOUNT
                RTS
```

D0 μετρητής για >65 ετών
D1 μετρητής επαναλήψεων

Πρόγραμμα 5ο

Κατά την κατασκευή ενός συστήματος συναγερμού που ελέγχεται από μικροεπεξεργαστή πρόκειται να γραφτεί μια υπορουτίνα που θα απενεργοποιεί το σύστημα όταν από το πληκτρολόγιο εισαχθεί μια σειρά έξι χαρακτήρων ASCII που αποτελούν των κωδικό απενεργοποίησης του συστήματος.

Οι χαρακτήρες αυτοί αποθηκεύονται στις θέσεις μνήμης $400401-$400406.

Το σύστημα θα απενεργοποιείται μόνο όταν οι χαρακτήρες αυτοί θα ταιριάζουν με μία σειρά χαρακτήρων ASCII που αποθηκεύονται αυτόματα από το σύστημα, όταν αυτό ενεργοποιείται, στις θέσεις $400407-$40040C της μνήμης.

Όταν ο συναγερμός είναι ενεργός στη θέση μνήμης $400400 είναι αποθηκευμένος ο αριθμός $FF και απενεργοποιείται όταν το περιεχόμενο της θέσης αυτής γίνει $00.

```
                ORG         $400410
SUBROUTINE:     CLR.L       D0
                LEA         $400401,A0
                LEA         $400407,A1
LOOP:           CMP         (A0)+,(A1)+
                BNE         ALARM_ON
                ADDQ.B      #1,D0
                CMPI.B      #6,D0
                BNE         LOOP
ALARM_OFF:      MOVE.B      #0,$400400
                RTS
ALARM_ON:       MOVE.B      #FF,$400400
                RTS
```

Πρόγραμμα 6ο

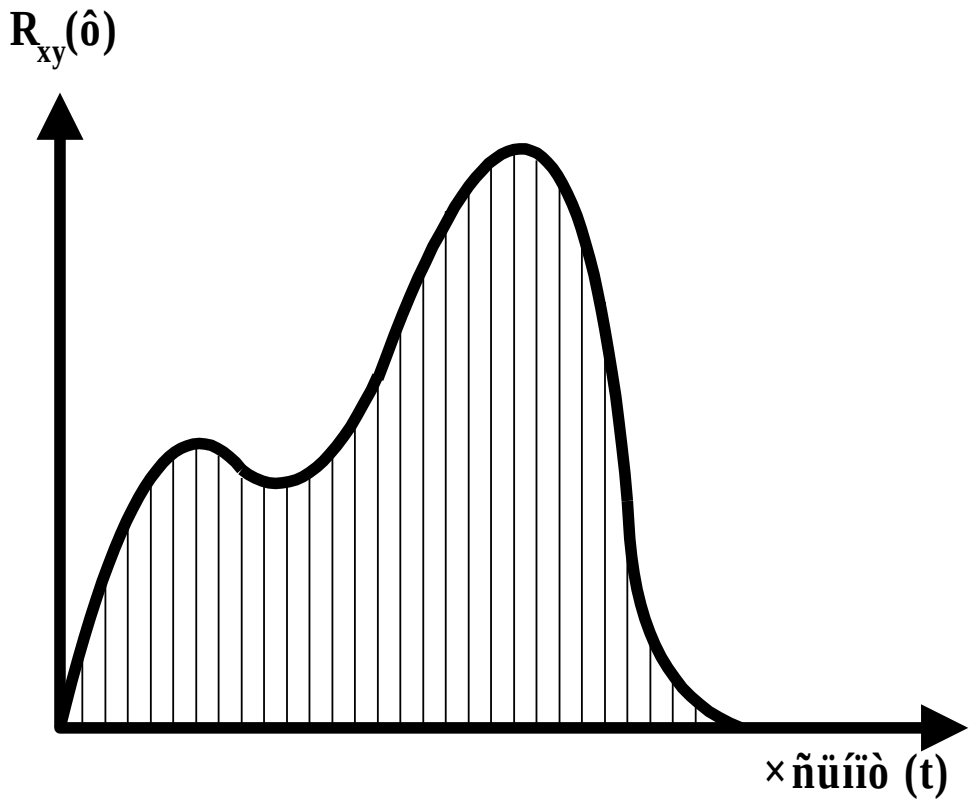
Σε μια χημική βιομηχανία υπάρχουν τέσσερα ρεζερβουάρ που περιέχουν θερμαινόμενα υγρά. Αισθητήρια στάθμης υγρού χρησιμοποιούνται στα ρεζερβουάρ A και B για να δείξουν το αν η στάθμη του υγρού έχει ανέβει πάνω από ένα προκαθορισμένο όριο. Αισθητήρια θερμοκρασίας χρησιμοποιούνται στα ρεζερβουάρ B και C για να δείξουν το αν η θερμοκρασία του υγρού έχει πέσει κάτω από κάποιο προκαθορισμένο όριο. Οι καταστάσεις των αισθητηρίων διαβάζονται συνεχώς και αντικατοπτρίζονται στα ψηφία d5 των θέσεων $400400 και $400401 αντίστοιχα για τα A και B και στα ψηφία d7 των θέσεων $400402 και $400403 αντίστοιχα για τα C και D. Αν υποτεθεί ότι η έξοδος των αισθητηρίων στάθμης είναι λογικό ¨0¨ όταν η στάθμη είναι ικανοποιητική και λογικό ¨1¨ όταν η στάθμη είναι υψηλή καθώς επίσης αν υποτεθεί ότι η έξοδος των αισθητηρίων θερμοκρασίας είναι λογικό ¨0¨ όταν η θερμοκρασία είναι ικανοποιητική και λογικό ¨1¨ όταν η θερμοκρασία είναι πολύ χαμηλή.

Να γραφτεί μια πρόγραμμα που θα ανιχνεύει την κατάσταση όπου η στάθμη στα ρεζερβουάρ A ή B είναι πολύ υψηλή ενώ ταυτόχρονα η θερμοκρασία στα ρεζερβουάρ C ή D είναι πολύ χαμηλή και θα ειδοποιεί κάνοντας το ψηφίο d1 της θέσης μνήμης $400404 λογικό “1”.

```
        ORG     $400410
TESTA:  MOVE.B  $400400,D0
        BTST    #5,D0
        BNE     TESTC
TESTB:  MOVE.B  $400401,D0
        BTST    #5,D0
        BNE     TESTC
        BRA     TESTA
TESTC:  MOVE.B  $400402,D0
        BTST    #7,D0
        BNE     ALARM
TESTD:  MOVE.B  $400403,D0
        BTST    #7,D0
        BNE     ALARM
        BRA     TESTB
ALARM:  ORI.B   #02,$400404
        BRA     TESTA
```

Πρόγραμμα 7ο

Στο σχήμα 1 φαίνεται μία ενδεικτική καμπύλη ετεροσυσχέτισης δύο σημάτων θορύβου που μεταφέρουν πληροφορία. Η πληροφορία αυτή θα είναι χρήσιμη στο βαθμό που θα είναι δυνατός ο εντοπισμός της κορυφής της καμπύλης. Η πληροφορία που αφορά στην καμπύλη παίρνεται από έναν συσχετιστή και αποθηκεύεται στις θέσεις μνήμης από $400500-$40051F, όπως φαίνεται στον πίνακα.



Πίνακας

400500	400501	400502	400503	400504	400505	400506	400507
00	27	35	41	5D	61	6E	60
400508	400509	40050A	40050B	40050C	40050D	40050E	40050F
B0	B9	C7	D1	E5	E0	D2	C3
400510	400511	400512	400513	400514	400515	400516	400517
B7	B1	A0	94	82	7A	6E	60
400518	400519	40051A	40051B	40051C	40051D	40051E	40051F
5F	52	4D	40	37	30	20	00

Ζητείται: *Να γραφτεί ένα πρόγραμμα που θα εντοπίζει την κορυφή της καμπύλης και θα αποθηκεύει το αποτέλεσμα στη θέση μνήμης $400520.*

```
        ORG     $400400

MAX     MOVE.B  #32,D0
        MOVEA.L $400500,A0
        CLR.L D1
LOOP    MOVE.B  (A0)+,D2
        CMP.B   D2,D1
        BCC     SMALLER
        MOVE.B  D2,D1
SMALLER SUBQ    #1,D0
NOCHG   BNE     LOOP
        MOVE.B  D1,$400520
        RTS
```

Πρόγραμμα 8ο

Σε ένα ερευνητικό κέντρο καταγράφουν τη θερμοκρασία κάθε ώρα της ημέρας χρησιμοποιώντας ένα byte για κάθε τιμή. Στο τέλος του χρόνου θέλουν να γνωρίζουν τη μέση θερμοκρασία έτους. Ο ορμαθός ωριαίων θερμοκρασιών είναι αποθηκευμένος απ' τη θέση μνήμης $400500 και πάνω. Να γραφτεί μια υπορουτίνα που θα υπολογίζει τη μέση θερμοκρασία και θα την αποθηκεύει σε μορφή ακέραιου αριθμού στη θέση μνήμης $400400.

```
        ORG     $400410
SUBRTN: LEA     $400500,A0
        MOVE.W  #8760,D0
        CLR.L   D1
        CLR.L   D2
LOOP:   MOVE.B  (A0)+,D1
        ADD.L   D1,D2
        SUBI.W  #1,D0
        BNE     LOOP
        MOVE.W  #8760,D0
        DIVU    D0,D2
        MOVE.W  D2,$400400
        RTS
```

Πρόγραμμα 9ο

Να γραφτεί μια υπορουτίνα που θα συγκρίνει δύο ορμαθούς byte μήκους 100 byte, που είναι αποθηκευμένοι αντίστοιχα απ' τις θέσεις $400500 και $400600 και πάνω. Κάθε φορά που η υπορουτίνα βρίσκει άνισα byte θα τα αποθηκεύει σε δύο άλλους ορμαθούς που αρχίζουν αντίστοιχα απ' τις θέσεις μνήμης $400700 και $400800. Αν οι ορμαθοί είναι ίδιοι θα μηδενίζει τη θέση μνήμης $400400.

```
        ORG     $400410
SUBRTN: LEA     $400500,A0
        LEA     $400600,A1
        LEA     $400700,A2
        LEA     $400800,A3
        CLR.B   D3
        MOVE    #100,D0
LOOP:   MOVE.B  (A0)+,D1
        MOVE.B  (A1)+,D2
        CMP.B   D1,D2
        BEQ     NEXT
        ADDQ.B  #1,D3
        MOVE.B  D1,(A2)+
        MOVE.B  D2,(A3)+
NEXT:   SUBQ.B  #1,D0
        BNE     LOOP
        CMPI.B  #0,D3
        BNE     FIN
        MOVE.B  #0,$400400
FIN:    RTS
```

Πρόγραμμα 10ο

Όταν χρησιμοποιείται άρτια ισοτιμία το ψηφίο ισοτιμίας που αποστέλλεται μαζί με την ψηφιολέξη των δεδομένων έχει τέτοια τιμή ώστε το σύνολο των ¨1¨ της ψηφιολέξης που αποστέλλεται να είναι άρτιος αριθμός. Το ψηφίο ισοτιμίας τοποθετείται αμέσως πριν απ' το περισσότερο σημαντικό ψηφίο της ψηφιολέξης δεδομένων. Πρόκειται να αποσταλεί ένας ορμαθός 1024 χαρακτήρων ASCII (εφτά ψηφία) που αρχίζει απ' τη θέση μνήμης $400500. Να γραφτεί μια υπορουτίνα που θα προσθέτει σε κάθε χαρακτήρα ASCII το αντίστοιχο ψηφίο ισοτιμίας και θα σχηματίσει ένα νέο ορμαθό, που τελικά θα εκπεμφθεί, και ο οποίος θα αποθηκεύεται στη μνήμη αρχίζοντας απ' τη θέση μνήμης αμέσως μετά τον τελευταίο χαρακτήρα του πρώτου ορμαθού.

```
SUBRTN:  LEA      $400500,A0
         LEA      $400900,A1
         MOVE.W   #1024,D0
LOOP:    MOVE.B   #7,D1
         CLR.L    D3
         MOVE.B   (A0)+,D2
         MOVE.B   D2,D6
LOOP1:   LSR.B    #1,D2
         BCC      ZERO
         ADDQ.B   #1,D3
ZERO:    SUBQ.B   #1,D1
         BNE      LOOP1
         DIVU     #2,D3
         SWAP     D3
         CMPI.W   #0,D3
         BEQ      ZYGOS
         ORI.B    #$80,D6
ZYGOS:   MOVE.B   D6,(A1)+
         SUBI.B   #1,D0
         BNE      LOOP
         RTS
```

Πρόγραμμα 11ο

Στο Τμήμα Βιομηχανικής Πληροφορικής είναι εγγεγραμμένοι 964 φοιτητές, που είναι καταχωρημένοι απ' τη θέση μνήμης $400100 και πάνω. Κάθε φοιτητής είναι καταχωρημένος με έναν κωδικό μήκους 16 ψηφίων. Κάθε φορά που ένας φοιτητής εγγράφεται ή ανανεώνει την εγγραφή του το ψηφίο d_{12} του κωδικού του παίρνει την τιμή "1". Να γραφεί ένα πρόγραμμα που θα κάνει αυτόματα τον έλεγχο εγγραφής του ή μη κάθε εξάμηνο. Αν είναι εγγεγραμμένος ο κωδικός του θα αποθηκεύεται σε μια νέα λίστα που θα αρχίζει απ' τη θέση $400500. Αν δεν είναι εγγεγραμμένος θα διαγράφεται απ' τα μητρώα του τμήματος και θα καταχωρείται σε μια άλλη λίστα που θα αρχίζει απ' τη θέση μνήμης $400900.

```
        ORG     $400400
SUBRTN: MOVE.W  #964,D0     * D0 μετρητής σπουδαστών.
        LEA     $400100,A0  * A0 δείκτης λίστας φοιτητών.
        LEA     $400500,A1  * A1 δείκτης νέας λίστας φοιτητών.
        LEA     $400900,A2  * A2 δείκτης λίστας διαγραφ. φοιτητών.
LOOP:   MOVE.W  (A0)+,D1    * Πάρε κωδικό επόμενου φοιτητή.
        ANDI.W  #1000,D1    * Έλεγξε το ψηφίο d12.
        BEQ     YES         * Είναι "1". Αν ναι, κάνε επανεγγραφή.
        MOVE.W  D1,(A2)+    * Διαφορετικά, διάγραψέ τον.
        BRA     NEXT        * Πήγαινε στον επόμενο.
YES:    MOVE.W  D1,(A1)+    * Βάλε τον στη νέα λίστα των εγγεγραμ.
NEXT:   SUBQ.W  #1,D0       * Έχουν ελεγχθεί όλοι οι φοιτητές
        BNE     LOOP        * Αν όχι. Συνέχισε έλεγχο.
        RTS                 * Επίστρεψε απ' την υπορουτίνα.
```

Πρόγραμμα 12ο

Να γραφτεί μια υπορουτίνα που θα εξομοιώνει τη λειτουργία ενός ρολογιού που λειτουργεί σε 24η βάση. Οι ώρες, τα λεπτά και τα δευτερόλεπτα θα αποθηκεύονται στις θέσεις μνήμης $400400, $400401 και $400402 αντίστοιχα.

```
            ORG     $400400
CLOCK:      CLR.B   $400400
            CLR.B   $400401
            CLR.B   $400402
LOOP:       CLR.B   D0
            CLR.B   D1
            CLR.B   D2
            MOVEQ   #24,D3
LOOP3       MOVEQ   #60,D4
LOOP2:      MOVEQ   #60,D5
LOOP1:      JSR     DELAY1sec
            ADDQ.B  #1,D2
            MOVEQ   D2,$400402
            SUBQ.B  #1,D5
            BNE     LOOP1
            CLR.B   $400402
            ADDQ.B  #1,D1
            MOVEQ   D1,$400401
            SUBQ.B  #1,D4
            BNE     LOOP2
            CLR.B   $400401
            ADDQ.B  #1,D0
            MOVEQ   D0,$400400
            SUBQ.B  #1,D3
            BNE     LOOP3
            CLR.B   $400400
            BRA     LOOP
DELAY1sec:  MOVE.L  #555553,D6
LOOP4:      SUBQ.L  #1,D6
            BNE     LOOP2
            RTS
```

Πρόγραμμα 13ο

Να γραφτεί μια υπορουτίνα που θα μετατρέπει τη λέξη που είναι αποθηκευμένη στη θέση μνήμης $400402 σε μια σειρά από δυαδικά ψηφία και θα τα αποθηκεύει στις θέσεις μνήμης $400404-$400409 με το λιγότερο σημαντικό ψηφίο στη θέση $400409.

Σημείωση: Ο μέγιστος δεκαδικός αριθμός που μπορεί να σχηματιστεί με μια ψηφιολέξη 16 δυαδικών ψηφίων είναι 65.535

```
                ORG     $400400
                DC.L    $0000FFFF
                DS.B    5
                ORG     $400410
SUBRTN:         LEA     $400409,A0
                MOVE.L  $400400,D0
                MOVEQ   #5,D1
LOOP:           DIVU    #10,D0
                SWAP    D0
                MOVE.B  D0,-(A0)
                CLR.B   D0
                SWAP    D0
                SUBQ.B  #1,D1
                BNE     LOOP
                RTS
```

Πρόγραμμα 14ο

Σ' ένα μεγάλο αεροδρόμιο καταγράφεται καθημερινά ο αριθμός των επιβατών που διακινούνται. Στο τέλος κάθε χρόνου ο αερολιμενικός έλεγχος πρέπει να βγάλει το μέσο ημερήσιο καθώς και το σύνολο των επιβατών που διακινήθηκαν. Αν υποτεθεί ότι οι ημερήσιοι αριθμοί επιβατών είναι καταχωρημένοι απ' τη θέση $400500 και πάνω. Να γραφτεί μια υπορουτίνα που θα βγάζει: α. Το σύνολο των επιβατών που διακινήθηκαν και β. Τη μέση ημερήσια διακίνηση και θα τα αποθηκεύει από τις θέσεις μνήμης $400400 και πάνω.

Σημείωση: Ο μέγιστος αριθμός διακινουμένων την ημέρα ήταν 12.167 επιβάτες.

```
                ORG         $400400
                DS.L        1
                DS.W        1
                ORG         $400410
SUBRTN:         MOVE.W      #365,D0
                CLR.L       D1
                LEA         $400500,A0
LOOP:           ADD.L       (A0)+,D1
                SUBQ.W      #1,D0
                BNE         LOOP
                MOVE.L      D1,$400400
LOOP:           MOVE.W      #365,D0
                DIVU        D0,D1
                MOVE.W      D1,$400404
                RTS
```

Πρόγραμμα 15ο

Ένα σύστημα αυτόματου ποτίσματος αποτελείται από τρία αισθητήρια υγρασίας Y_2, Y_1, Y_0 των οποίων οι καταστάσεις αντικατοπτρίζονται στα ψηφία $d_2d_1d_0$ της θέσης μνήμης $400400 και τρεις ηλεκτροβάνες ποτίσματος H_2, H_1, H_0, που ποτίζουν τρεις ποικιλίες διαφορετικών φυτών με διαφορετικές ανάγκες σε νερό, και των οποίων οι καταστάσεις εξαρτώνται από τα ψηφία $d_2d_1d_0$ αντίστοιχα της θέσης μνήμης $400401. Να γραφτεί ένα πρόγραμμα αυτόματου ποτίσματος που θα ενεργοποιεί (λογικό 1):

α. τη βάνα H_0 για 30min κάθε φορά που το αισθητήριο Y_0 δείχνει χαμηλή υγρασία (λογικό 1).

β. τη βάνα H_1 για 20min κάθε φορά που το αισθητήριο Y_1 δείχνει χαμηλή υγρασία (λογικό 1).

γ. τη βάνα H_2 για 10min κάθε φορά που το αισθητήριο Y_2 δείχνει χαμηλή υγρασία (λογικό 1).

Σημείωση: Η βάνα ενεργοποιείται με λογικό 1 και μόνο μία βάνα μπορεί να είναι ενεργοποιημένη κάθε φορά.

```
        ORG     $400402
SUBRTN  MOVE.B  $400400,D0
        BTST    #0,D0
        BEQ     CHECK1
        MOVEQ   #1,$400401
        MOVE.W  #1800,D2
        BSR     DELAY
CHECK1  BTST    #1,D0
        BEQ     CHECK2
        MOVEQ   #2,$400401
        MOVE.W  #1200,D2
        BSR     DELAY
CHECK2  BTST    #2,D0
        BEQ     SUBRTN
        MOVEQ   #4,$400401
        MOVE.W  #600,D2
        BSR     DELAY
        BRA     SUBRTN
DELAY   MOVE.L  #555553,D6
DEL1    SUBQ.L  #1,D6
        BNE     DEL1
        SUBQ.W  #1,D2
        BNE     DELAY
        MOVE.B  #0,$400401
        RTS
```

Πρόγραμμα 16ο

Να γραφτεί μια υπορουτίνα που θα ολισθαίνει προς τα δεξιά έναν προσημασμένο αριθμό πολλών byte που είναι αποθηκευμένος στη μνήμη, με το λιγότερο σημαντικό byte να ξεκινά από την θέση μνήμης $400501. Το μήκος του αριθμού σε byte είναι αποθηκευμένο στη θέση μνήμης $400500.

```
SUBRTN    MOVE.B    $400500,D0
          MOVEA.L   $400500,A0
          ADD.B     D0,A0
          MOVE.L    A0,A1
          MOVE.B    -(A0),D1
          ASR.B     #1,D1
          MOVE.B    D1,-(A1)
          SUBQ.B    #1,D0
          BEQ       FIN
LOOP      MOVE.B    -(A0),D1
          ROXR.B    #1,D1
          MOVE.B    D1,-(A1)
          SUBQ.B    #1,D0
          BNE LOOP
FIN:      RTS
```