

# Μάθημα 2.1

## Συστήματα Αρίθμησης

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τον τρόπο παράστασης των αριθμών σε διάφορα συστήματα αρίθμησης πέρα από το γνωστό μας δεκαδικό.

Σκοπός του μαθήματος

Όταν ολοκληρώσεις το μάθημα αυτό θα μπορείς:

- ♦ Να εξηγείς πώς παριστάνονται οι αριθμοί στα διάφορα συστήματα αρίθμησης.
- ♦ Να μετατρέπεις αριθμούς από και προς διάφορα συστήματα αρίθμησης.

Τι θα μάθεις;

Οι υπολογιστές αναπαριστούν όλα τα είδη πληροφορίας ως δυαδικά δεδομένα. Έτσι, για την ευκολότερη και ταχύτερη επεξεργασία των διαφόρων πληροφοριών, οι υπολογιστές χρησιμοποιούν αριθμητικά συστήματα διαφορετικά από το γνωστό μας δεκαδικό (decimal) σύστημα και κυρίως το δυαδικό (binary).

### Αριθμητικά Συστήματα

Κάθε αριθμός  $N$  μπορεί να γραφεί με την ακόλουθη μορφή:

$$N = \sum_{i=-n}^{m-1} \alpha_i \cdot \beta^i = \underbrace{\alpha_{m-1} \cdot \beta^{m-1} + \alpha_{m-2} \cdot \beta^{m-2} + \dots + \alpha_1 \cdot \beta^1 + \alpha_0 \cdot \beta^0}_{\text{ακέραιο μέρος του αριθμού}} + \underbrace{\alpha_{-1} \cdot \beta^{-1} + \alpha_{-2} \cdot \beta^{-2} + \dots + \alpha_{-n} \cdot \beta^{-n}}_{\text{κλασματικό μέρος του αριθμού}}$$

Με  $\alpha_i$  συμβολίζουμε τα ψηφία του αριθμού και με  $\beta$  παριστάνουμε τη βάση του αριθμητικού συστήματος στο οποίο εκφράζεται ο αριθμός. Το ψηφίο  $\alpha_i$  πολλαπλασιάζεται με τον αριθμό  $\beta^i$ , γι' αυτό λέμε ότι η τάξη (order) του ψηφίου  $\alpha_i$  είναι  $i$ . Το κλασματικό τμήμα του αριθμού είναι αυτό μετά την υποδιαστολή και είναι μικρότερο του 1. Αν ο αριθμός  $N$  έχει  $m$  ακέραια ψηφία, οι εκθέτες  $i$  παίρνουν τιμές από 0 έως  $m-1$  για το ακέραιο μέρος του και αν τα κλασματικά του ψηφία είναι  $n$ , οι εκθέτες  $i$  παίρνουν αρνητικές τιμές από  $-1$  έως  $-n$  για το κλασματικό του τμήμα.

Ο δεκαδικός αριθμός 19,278 με τον τρόπο αυτό γράφεται ως  $1 \cdot 10^1 + 9 \cdot 10^0 + 2 \cdot 10^{-1} + 7 \cdot 10^{-2} + 8 \cdot 10^{-3}$ , δηλαδή  $\alpha_1=1$ ,  $\alpha_0=9$ ,  $\alpha_{-1}=2$ ,  $\alpha_{-2}=7$ ,  $\alpha_{-3}=8$ .

Ένα αριθμητικό σύστημα με βάση  $\beta$  χρειάζεται  $\beta$  διαφορετικά «ψηφία» για την παράσταση των αριθμών, που παίρνουν τις τιμές από 0 έως  $\beta-1$ . Ένας φυσικός αριθμός που έχει  $m$  ψηφία, στο σύστημα αυτό μπορεί να πάρει τιμές από 0 έως  $\beta^m-1$ , δηλαδή  $\beta^m$  διαφορετικές τιμές.

Δεκαδικά ψηφία	Δυαδικά ψηφία	Οκταδικά ψηφία	Δεκαεξαδικά ψηφία
0	0	0	0
1	1	1	1
2		2	A
3		3	B
4			C
			D
			E
			F

Τα συνηθέστερα αριθμητικά συστήματα είναι αυτά που έχουν βάση τους αριθμούς 2 (δυαδικό σύστημα), 8 (οκταδικό σύστημα, octal system), 10 (δεκαδικό σύστημα) και 16 (δεκαεξαδικό σύστημα, hexadecimal system). Στον πίνακα βλέπουμε τα ψηφία αυτών των αριθμητικών συστημάτων.

Το δεκαεξαδικό σύστημα χρειάζεται 16 ψηφία για την παράσταση των αριθμών, αλλά το γνωστό μας αριθμητικό σύστημα παρέχει μόνο 10 ψηφία (0-9). Για τα επιπλέον 6

ψηφία χρησιμοποιούμε τους χαρακτήρες A-F, δηλαδή A=10, B=11, C=12, D=13, E=14 και F=15.

Στο δεκαεξαδικό σύστημα, η ακολουθία ψηφίων «6F3» παριστάνει τον αριθμό  $6 \cdot 16^2 + 15 \cdot 16^1 + 3 \cdot 16^0$

Στην καθημερινή μας ζωή χρησιμοποιούμε το δεκαδικό σύστημα και είμαστε εξοικειωμένοι με αυτό, έτσι παριστάνουμε τους αριθμούς μόνο με τα ψηφία τους, π.χ. λέμε 15 και όχι  $1 \cdot 10^1 + 5 \cdot 10^0$ . Το ίδιο μπορούμε να κάνουμε και με οποιοδήποτε άλλο αριθμητικό σύστημα, αρκεί να δηλώνουμε το σύστημα αυτό. Ο προσδιορισμός του συστήματος γίνεται συνήθως με ένα δείκτη που συνοδεύει τον αριθμό και δηλώνει τη βάση του αριθμητικού συστήματος.

Η ακολουθία ψηφίων «321» παριστάνει διαφορετικούς αριθμούς ανάλογα με το σύστημα αρίθμησης που θα δηλώσουμε. Στο δεκαδικό σύστημα θα γράψουμε  $321_{(10)}$  και θα εννοούμε  $3 \cdot 10^2 + 2 \cdot 10^1 + 1 \cdot 10^0$ , ενώ στο οκταδικό σύστημα θα γράψουμε  $321_{(8)}$  και θα εννοούμε  $3 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0$ . Στο δεκαεξαδικό σύστημα θα γράψουμε  $321_{(16)}$  και θα εννοούμε  $3 \cdot 16^2 + 2 \cdot 16^1 + 1 \cdot 16^0$ .

Ο αριθμός  $1101_{(2)}$  είναι γραμμένος στο δυαδικό σύστημα, έτσι αντιστοιχεί στην έκφραση  $1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$ .

Τα ψηφία ενός αριθμού γραμμένου στο δυαδικό σύστημα ονομάζονται bits (binary digits, δυαδικά ψηφία). Το πιο αριστερό ψηφίο του αριθμού ονομάζεται *περισσότερο σημαντικό ψηφίο* (Most Significant Bit, MSB), γιατί πολλαπλασιάζεται με το μεγαλύτερο συντελεστή, και το πιο δεξιό ψηφίο του αριθμού ονομάζεται *λιγότερο σημαντικό ψηφίο* (Least Significant Bit, LSB), γιατί πολλαπλασιάζεται με το μικρότερο συντελεστή.

## Μετατροπή αριθμών από ένα αριθμητικό σύστημα σε άλλο

Η μετατροπή ενός αριθμού από ένα αριθμητικό σύστημα με βάση  $\beta$  προς το δεκαδικό σύστημα είναι πολύ απλή: υπολογίζουμε την τιμή της παράστασης  $a_{m-1} \cdot \beta^{m-1} + \dots + a_1 \cdot \beta^1 + a_0 \cdot \beta^0 + a_{-1} \cdot \beta^{-1} + \dots + a_{-n} \cdot \beta^{-n}$ .

Ο δυαδικός αριθμός  $10011_{(2)}$  στο δεκαδικό σύστημα έχει την τιμή  $1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 16 + 2 + 1 = 19_{(10)}$ .

Ο οκταδικός αριθμός  $7123,35_{(8)}$  στο δεκαδικό σύστημα έχει την τιμή  $7 \cdot 8^3 + 1 \cdot 8^2 + 2 \cdot 8^1 + 3 \cdot 8^0 + 3 \cdot 8^{-1} + 5 \cdot 8^{-2} = 3584 + 64 + 16 + 3 + 0,375 + 0,0781 = 3667,4531_{(10)}$ .

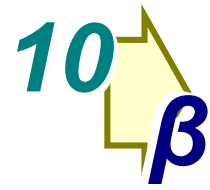
Ο δεκαεξαδικός αριθμός  $FC27_{(16)}$  αντιστοιχεί στο δεκαδικό  $15 \cdot 16^3 + 12 \cdot 16^2 + 2 \cdot 16^1 + 7 \cdot 16^0 = 61440 + 3072 + 32 + 7 = 64551_{(10)}$ .

Το δεκαδικό σύστημα δεν είναι το μόνο που έχει χρησιμοποιηθεί από τον άνθρωπο. Οι Βαβυλώνιοι χρησιμοποιούσαν αριθμητικό σύστημα με βάση το 60.



Πιο πολύπλοκη είναι η διαδικασία μετατροπής ενός αριθμού από το δεκαδικό σύστημα σε ένα άλλο σύστημα αρίθμησης με βάση  $\beta$ . Η μετατροπή γίνεται χωριστά για το ακέραιο και χωριστά για το κλασματικό μέρος.

Για να μετατρέψουμε το ακέραιο μέρος του αριθμού  $A$  σε βάση  $\beta$ , κάνουμε διαδοχικές διαιρέσεις του ακεραίου μέρους του  $A$  με τον αριθμό  $\beta$ . Η διαδικασία μετατροπής είναι η εξής:



- (1) Διαιρούμε το δεκαδικό αριθμό με το  $\beta$ , και παίρνουμε το πηλίκο  $\Pi$  και το υπόλοιπο  $\Upsilon$ . Το υπόλοιπο είναι μία τιμή από 0 έως  $\beta-1$  και αποτελεί το δεξιότερο ψηφίο του αριθμού.
- (2) Διαιρούμε το πηλίκο  $\Pi$  πάλι με το  $\beta$ , και παίρνουμε ένα νέο πηλίκο  $\Pi$  και υπόλοιπο  $\Upsilon$ . Γράφουμε το υπόλοιπο (που πάλι είναι μία τιμή από 0 έως  $\beta-1$ ) στα αριστερά του αριθμού.
- (3) Επαναλαμβάνουμε το βήμα (2) μέχρι το πηλίκο  $\Pi$  να γίνει 0.

Ας δούμε πώς μετατρέπεται ο αριθμός  $A=53_{(10)}$  στο δυαδικό σύστημα ( $\beta=2$ ):

		$\Pi$	$\Upsilon$	$X$
53	Διαιρούμε το 53 με το 2	26	1	1
26	Διαιρούμε το 26 με το 2	13	0	01
13	Διαιρούμε το 13 με το 2	6	1	101
6	Διαιρούμε το 6 με το 2	3	0	0101
3	Διαιρούμε το 3 με το 2	1	1	10101
1	Διαιρούμε το 1 με το 2	0	1	110101
0				$53_{(10)} = 110101_{(2)}$

Ας δούμε και τη μετατροπή του αριθμού 312 στο οκταδικό σύστημα:

		$\Pi$	$\Upsilon$	$X$
312	Διαιρούμε το 312 με το 8	39	0	0
39	Διαιρούμε το 39 με το 8	4	7	70
4	Διαιρούμε το 4 με το 8	0	4	470
0				$312_{(10)} = 470_{(8)}$

Για να μετατρέψουμε το κλασματικό μέρος ενός αριθμού  $A$  από το δεκαδικό σύστημα σε ένα άλλο σύστημα αρίθμησης με βάση  $\beta$ , κάνουμε διαδοχικούς πολλαπλασιασμούς του κλασματικού μέρους του  $A$  με τη βάση  $\beta$ . Το κλασματικό μέρος στο νέο σύστημα αρίθμησης μπορεί να έχει άπειρα ψηφία, γι' αυτό καθορίζουμε από πριν το μέγιστο αριθμό ψηφίων  $N$  που θα υπολογίσουμε για το νέο σύστημα αρίθμησης. Η διαδικασία μετατροπής είναι η ακόλουθη:

- (1) Χρησιμοποιούμε μόνο το κλασματικό μέρος του  $A$ .
- (2) Πολλαπλασιάζουμε τον  $A$  με τη βάση  $\beta$ . Το αποτέλεσμα έχει ακέραιο μέρος  $M$  και κλασματικό μέρος  $K$ . Το  $M$  (που έχει τιμή από 0 έως  $\beta-1$ ) είναι το αριστερότερο ψηφίο του νέου κλασματικού μέρους  $\Upsilon$ .

- (3) Πολλαπλασιάζουμε το κλασματικό μέρος  $K$  με το  $\beta$  και παίρνουμε ένα νέο αποτέλεσμα με ακέραιο μέρος  $M$  (με τιμή από 0 έως  $\beta-1$ ) και κλασματικό μέρος  $K$ .
- (4) Γράφουμε το  $M$  στα δεξιά του νέου κλασματικού μέρους  $Y$ .
- (5) Επαναλαμβάνουμε τα βήματα (3), (4) έως ότου το  $M$  να γίνει 0 ή να έχουμε υπολογίσει  $N$  ψηφία.

Ας υπολογίσουμε την τιμή του κλασματικού αριθμού  $0,625_{(10)}$  στο δυαδικό σύστημα:

			<b>M</b>	<b>K</b>	<b>Y</b>
<b>0,625</b>	—	$0,625 \times 2 = 1,25$	→ 1	0,25	0, <b>1</b>
0,25	—	$0,25 \times 2 = 0,5$	→ 0	0,5	0,1 <b>0</b>
0,5	—	$0,5 \times 2 = 1$	→ 1	0	0,10 <b>1</b>
0	→				<b><math>0,625_{(10)} = 0,101_{(2)}</math></b>

Επίσης ας υπολογίσουμε την τιμή του κλασματικού αριθμού  $0,171875_{(10)}$  στο δεκαξαδικό σύστημα:

			<b>M</b>	<b>K</b>	<b>Y</b>
<b>0, 171875</b>	—	$0,171875 \times 16 = 2,75$	→ 2	0,75	0, <b>2</b>
0,75	—	$0,75 \times 16 = 12$	→ 12	0	0,2 <b>C</b>
0	→				<b><math>0,171875_{(10)} = 0,2C_{(16)}</math></b>

Ας υπολογίσουμε και την τιμή του κλασματικού αριθμού  $0,4_{(10)}$  στο οκταδικό σύστημα. Καθορίζουμε από πριν ότι θα υπολογίσουμε το πολύ 5 οκταδικά ψηφία.

			<b>M</b>	<b>K</b>	<b>Y</b>
<b>0,4</b>	—	$0,4 \times 8 = 3,2$	→ 3	0,2	0, <b>3</b>
0,2	—	$0,2 \times 8 = 1,6$	→ 1	0,6	0,3 <b>1</b>
0,6	—	$0,6 \times 8 = 4,8$	→ 4	0,8	0,31 <b>4</b>
0,8	—	$0,8 \times 8 = 6,4$	→ 6	0,4	0,314 <b>6</b>
0,4	—	$0,4 \times 8 = 3,2$	→ 3	0,2	0,3146 <b>3</b>
0,2	→				<b><math>0,4_{(10)} \equiv 0,31463_{(8)}</math></b>

Εδώ σταματήσαμε τον υπολογισμό παρότι το  $M$  δεν είναι 0, γιατί φθάσαμε στο μέγιστο αριθμό κλασματικών ψηφίων που επιτρέπουμε. Αν προσέξουμε τις τιμές που παίρνει το  $M$ , παρατηρούμε ότι μετά από τον υπολογισμό του 4<sup>ου</sup> κλασματικού ψηφίου το  $M$  παίρνει την αρχική του τιμή που ήταν 0,4. Έτσι μπορούμε να συμπεράνουμε ότι στο οκταδικό σύστημα ο  $A$  είναι ένας περιοδικός αριθμός, ο  $0,31463146..._{(8)}$  ή αλλιώς  $0,\overline{3146}_{(8)}$ .

Όταν σταματάμε τον υπολογισμό του αριθμού μετά από  $N$  κλασματικά ψηφία, αν και το  $M$  δεν είναι 0, κάνουμε *αποκοπή* (truncation) του αριθμού. Για να είναι ο υπολογισμός μας όσο πιο ακριβής γίνεται, μπορούμε να κάνουμε *στρογγυλοποίηση* (rounding). Στη στρογγυλοποίηση υπολογίζουμε άλλο ένα κλασματικό ψηφίο. Αν αυτό είναι μικρότερο από το  $\frac{1}{2}$  της βάσης, τότε αφήνουμε τον αριθμό όπως είναι, με  $N$

Κατά τη μετατροπή του αριθμού  $0,4_{(10)}$  στο οκταδικό σύστημα, ολοκληρώσαμε τον υπολογισμό μετά από 5 ψηφία. Το επόμενο ψηφίο που θα υπολογίζαμε είναι το 1, που είναι μικρότερο από το 4 (το  $\frac{1}{2}$  της βάσης), έτσι και μετά από τη στρογγυλοποίηση ο αριθμός μένει ο ίδιος.

Αν όμως κρατούσαμε μόνο  $N=3$  κλασματικά ψηφία στρογγυλοποιώντας το αποτέλεσμα, θα υπολογίζαμε και το 4<sup>ο</sup> ψηφίο που έχει την τιμή  $6 > 4$ . Θα αυξάναμε λοιπόν το 3ο ψηφίο κατά 1, και ο αριθμός θα ήταν τελικά ο  $0,315_{(8)}$ .

Για να μετατρέψουμε από το δεκαδικό σύστημα αρίθμησης σε άλλο έναν αριθμό που έχει και ακέραιο και κλασματικό μέρος, μετατρέπουμε ξεχωριστά τα δύο μέρη του με τον τρόπο που είδαμε και μετά συνδυάζουμε τα αποτελέσματα.

Η μετατροπή ενός αριθμού από ένα σύστημα με βάση  $\beta_1$  σε ένα άλλο σύστημα με βάση  $\beta_2$  γίνεται εύκολα αν χρησιμοποιήσουμε ενδιάμεσα το δεκαδικό σύστημα: μετατρέπουμε πρώτα τον αριθμό με βάση  $\beta_1$  στο δεκαδικό σύστημα, και στη συνέχεια τον μετατρέπουμε από το δεκαδικό σύστημα στο σύστημα με βάση  $\beta_2$ . Η μέθοδος αυτή είναι πιο εύκολη από την απευθείας μετατροπή, γιατί είμαστε πιο εξοικειωμένοι με υπολογισμούς στο δεκαδικό σύστημα.

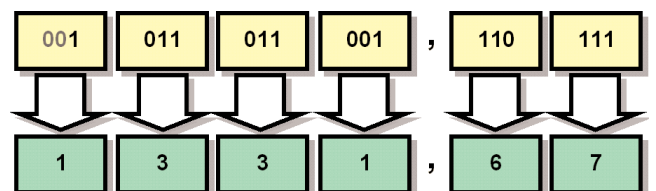
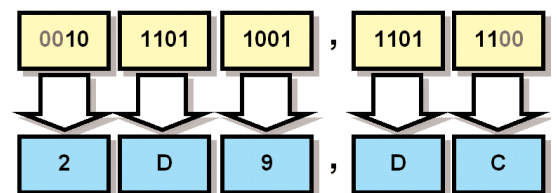
Μια ειδική περίπτωση όμως είναι η μετατροπή μεταξύ του δυαδικού και του οκταδικού ή του δεκαεξαδικού συστήματος. Οι μετατροπές αυτές είναι ιδιαίτερα εύκολες, γιατί οι βάσεις των δύο συστημάτων, το 8 και το 16, είναι δυνάμεις του 2.

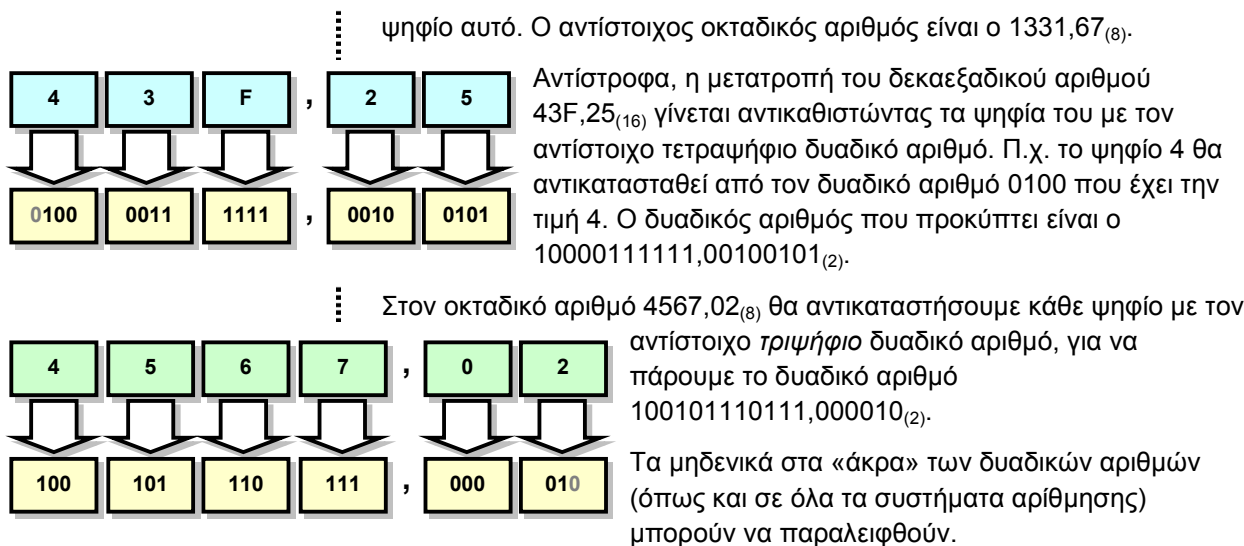
Για να μετατρέψουμε ένα δυαδικό αριθμό στο δεκαεξαδικό σύστημα, χωρίζουμε τα ψηφία του σε τετράδες ξεκινώντας από την υποδιαστολή που χωρίζει ακέραιο και κλασματικό μέρος, και προχωρώντας προς τα «άκρα» του αριθμού. Κάθε τέτοια τετράδα αντιστοιχεί σε ένα μονοψήφιο δεκαεξαδικό αριθμό, και την αντικαθιστούμε με το ψηφίο αυτό. Η μετατροπή ενός δεκαεξαδικού αριθμού σε δυαδικό γίνεται αντικαθιστώντας κάθε ψηφίο του αριθμού με τον αντίστοιχο τετραψήφιο δυαδικό αριθμό.

Η μετατροπή από το δυαδικό σύστημα προς το οκταδικό και αντίστροφα γίνεται με τον ίδιο τρόπο, αλλά χωρίζουμε τα δυαδικά ψηφία σε τριάδες αντί για τετράδες.

Για να μετατραπεί σε δεκαεξαδικό, χωρίζουμε το δυαδικό αριθμό  $1011011001,110111_{(2)}$  πρώτα σε τετράδες ξεκινώντας από την υποδιαστολή. Στα «άκρα» του αριθμού προσθέτουμε όσα μηδενικά είναι απαραίτητα, έτσι ώστε να συμπληρωθούν οι τετράδες. Στη συνέχεια αντικαθιστούμε κάθε τετράδα με το αντίστοιχο δεκαεξαδικό ψηφίο. Π.χ. η αριστερότερη τετράδα που είναι 0010 ισοδυναμεί με το ψηφίο 2, ενώ η επόμενη τετράδα 1101 που έχει τιμή 13 ισοδυναμεί με το ψηφίο D. Ο ισοδύναμος δεκαεξαδικός αριθμός είναι ο  $2D9,DC_{(16)}$ .

Για να μετατρέψουμε τον ίδιο αριθμό σε οκταδικό τον χωρίζουμε σε τριάδες, προσθέτοντας και πάλι στα άκρα του μηδενικά, αν χρειαστεί. Κάθε τριάδα αντικαθίσταται από το αντίστοιχο οκταδικό ψηφίο. Π.χ. η δεξιότερη τριάδα που είναι 111 έχει την τιμή 7 ( $1 \cdot 2^2 + 1 \cdot 2 + 1 = 4 + 2 + 1 = 7$ ) και αντικαθίσταται με το ψηφίο αυτό. Ο αντίστοιχος οκταδικός αριθμός είναι ο  $1331,67_{(8)}$ .





Ένα αριθμητικό σύστημα με βάση  $\beta$  παριστάνει κάθε αριθμό ως άθροισμα δυνάμεων του αριθμού  $\beta$ . Τα πιο γνωστά συστήματα είναι το δυαδικό, το οκταδικό, το δεκαδικό και το δεκαεξαδικό. Η μετατροπή ενός ακέραιου ή πραγματικού αριθμού από το δεκαδικό σύστημα σε ένα άλλο σύστημα με βάση  $\beta$  γίνεται με διαδοχικές διαιρέσεις ή πολλαπλασιασμούς με τη βάση  $\beta$ , αντίστοιχα. Οι μετατροπές μεταξύ άλλων συστημάτων αρίθμησης γίνονται για ευκολία μέσω του δεκαδικού συστήματος. Η μόνη εξαίρεση είναι οι μετατροπές μεταξύ δυαδικού και οκταδικού ή δεκαεξαδικού συστήματος, που γίνονται πολύ εύκολα απευθείας.



Αποκοπή	Truncation
Δεκαδικό Σύστημα	Decimal System
Δεκαεξαδικό Σύστημα	Hexadecimal System
Δυαδικό Σύστημα	Binary System
Δυαδικό Ψηφίο	Binary Digit - Bit
Λιγότερο Σημαντικό Ψηφίο	Least Significant Bit - LSB
Οκταδικό Σύστημα	Octal System
Περισσότερο Σημαντικό Ψηφίο	Most Significant Bit - MSB
Στρογγυλοποίηση	Rounding
Τάξη	Order

## Ερωτήσεις

- ? Πόσα ψηφία χρειάζεται ένα σύστημα αρίθμησης με βάση  $\beta$ ;
- ? Πώς μετατρέπουμε έναν ακέραιο δυαδικό αριθμό στο δεκαδικό σύστημα;
- ? Πώς μετατρέπουμε ένα δεκαδικό αριθμό στο δυαδικό σύστημα;
- ? Τι είναι η αποκοπή και η στρογγυλοποίηση;
- ? Πώς γίνεται η μετατροπή μεταξύ δυαδικού και οκταδικού συστήματος;



## Μάθημα 2.2

## Παράσταση Αριθμών και Χαρακτήρων

Σκοπός του μαθήματος αυτού είναι να περιγράψει πώς παριστάνονται οι αριθμοί σε ένα υπολογιστή και τη διαδικασία με την οποία γίνονται οι αριθμητικές πράξεις μεταξύ τους.

Σκοπός του μαθήματος

Όταν ολοκληρώσεις το μάθημα αυτό θα μπορείς:

- ♦ Να εξηγείς πώς γίνονται οι αριθμητικές πράξεις μεταξύ ακεραίων αριθμών
- ♦ Να υπολογίζεις την παράσταση προσημασμένων ακεραίων αριθμών και να εκτελείς πράξεις μεταξύ τους
- ♦ Να υπολογίζεις την παράσταση πραγματικών αριθμών και να εκτελείς πράξεις μεταξύ τους
- ♦ Να βρίσκεις την παράσταση χαρακτήρων με την κωδικοποίηση ASCII

Τι θα μάθεις;

### Αριθμητικές πράξεις

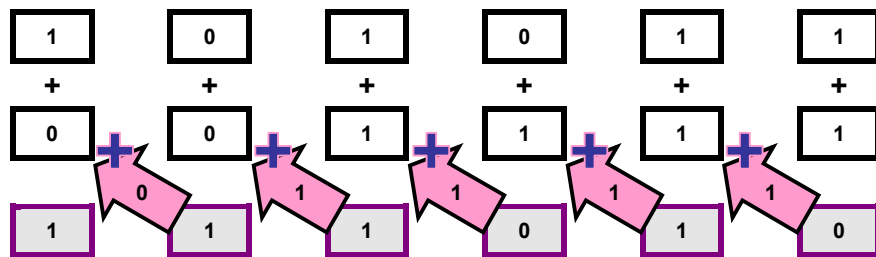
Στο δυαδικό σύστημα οι αριθμητικές πράξεις γίνονται όπως και στο δεκαδικό σύστημα. Προσθέσεις και αφαιρέσεις γίνονται από δεξιά προς τα αριστερά, και χρησιμοποιούμε κρατούμενα ή δανεικά ψηφία αντίστοιχα.

Για την πρόσθεση αριθμών με  $n$  bits, ξεκινάμε από τα δεξιά, προσθέτοντας τα δύο λιγότερα σημαντικά bits των αριθμών και συνεχίζουμε προς τα αριστερά. Αν οι αριθμοί είναι οι  $x_n x_{n-1} \dots x_i \dots x_1$  και  $y_n y_{n-1} \dots y_i \dots y_1$ , ξεκινάμε από την πρόσθεση  $x_1 + y_1$ , η οποία μας δίνει το ψηφίο  $z_1$  του αποτελέσματος και το κρατούμενο  $K_1$ . Στη συνέχεια προσθέτουμε τα ψηφία  $x_2, y_2$  και το κρατούμενο  $K_1$  για να πάρουμε το ψηφίο  $z_2$  του αποτελέσματος και το κρατούμενο  $K_2$ , κλπ. Γενικά το κρατούμενο  $K_{i-1}$  που πιθανώς θα προκύψει από κάποια πρόσθεση προωθείται και προστίθεται με το αμέσως αριστερότερο ζεύγος ψηφίων  $x_i$  και  $y_i$  των αριθμών. Αντίστοιχα στην αφαίρεση λαμβάνουμε τα αντίστοιχα ζεύγη των ψηφίων  $x_i$  και  $y_i$ , κάνουμε την πράξη και προωθούμε το δανεικό ψηφίο  $\Delta_i$  στο επόμενο ζεύγος ψηφίων  $x_{i+1}$  και  $y_{i+1}$ .

Οι υπολογισμοί που κάνουμε για αριθμούς των  $n$  bits αφορούν λοιπόν συνήθως 3 bits, γιατί έχει προστεθεί και το κρατούμενο ή το δανεικό ψηφίο. Στον πίνακα βλέπουμε τα αποτελέσματα της πρόσθεσης και αφαίρεσης δύο ψηφίων με κρατούμενο ή δανεικό ψηφίο αντίστοιχα.

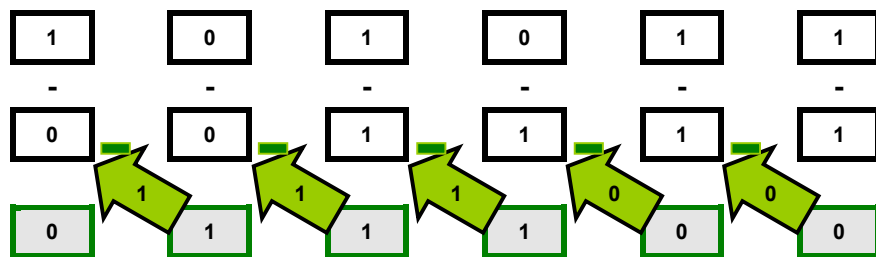
$x_i$	$y_i$	$K_{i-1}$ ή $\Delta_{i-1}$	$x_i + y_i + K_{i-1}$	$K_i$	$x_i - y_i - \Delta_{i-1}$	$\Delta_i$
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Θέλουμε να προσθέσουμε τους αριθμούς 43 και 15, αλλά με τη δυαδική τους αναπαράσταση. Ο αριθμός 43 στο δυαδικό σύστημα είναι ο  $101011_{(2)}$  ενώ ο 15 παριστάνεται ως  $001111_{(2)}$ .



Βλέπουμε στο σχήμα πώς το κρατούμενο από κάθε ζεύγος ψηφίων προωθείται στο επόμενο. Το αποτέλεσμα είναι ο δυαδικός αριθμός  $111010_{(2)}$  δηλαδή ο αριθμός 58.

Αν θέλουμε να κάνουμε την αφαίρεση  $43-15$  αντίστοιχα θα έχουμε:



Το αποτέλεσμα είναι ο δυαδικός αριθμός  $011100_{(2)}$  δηλαδή ο δεκαδικός 28. Και εδώ βλέπουμε τα δανεικά ψηφία του κάθε ζεύγους που μεταφέρονται στο επόμενο επίπεδο.

Ο πολλαπλασιασμός και η διαίρεση δυαδικών αριθμών γίνονται με διαδοχικές προσθέσεις και αφαιρέσεις αντίστοιχα.

**τελικό αποτέλεσμα**

Ας δούμε πώς πολλαπλασιάζουμε τους αριθμούς  $22_{(10)} = 10110_{(2)}$  και  $11_{(10)} = 1011_{(2)}$ .

Για να δημιουργήσουμε κάθε ένα από τα μερικά γινόμενα, πολλαπλασιάζουμε τον «πάνω» αριθμό (τον 10110) με το αντίστοιχο ψηφίο του άλλου αριθμού (του 1011) εργαζόμενοι από τα δεξιά προς τα αριστερά. Κάθε μερικό γινόμενο είναι μετακινημένο κατά μία θέση αριστερά σε σχέση με το προηγούμενο, ώστε η πρόσθεση όλων μαζί να μας δώσει το σωστό αποτέλεσμα, και έχει συμπληρωθεί από τα δεξιά με μηδενικά. Αυτή η μετακίνηση ονομάζεται *ολίσθηση προς τα αριστερά* (left shift) και ισοδυναμεί με πολλαπλασιασμό: ολίσθηση προς τα αριστερά κατά μία θέση είναι ισοδύναμη με πολλαπλασιασμό του αριθμού επί 2.



## Παράσταση αριθμών

Η μνήμη κάθε υπολογιστή είναι οργανωμένη σε λέξεις (words), δηλαδή ομάδες των  $n$  bits (το  $n$  είναι συνήθως ένα πολλαπλάσιο του 8). Το  $n$  ονομάζεται *μήκος λέξης* (word length) του υπολογιστή. Κάθε αριθμός θα καταλαμβάνει χώρο όσο μία λέξη της μνήμης του υπολογιστή.

Με  $N$  δυαδικά ψηφία μπορούμε να παραστήσουμε  $2^N$  το πλήθος διαφορετικούς αριθμούς, τους  $0 \dots 2^N - 1$ . Στην περίπτωση που θέλουμε με  $n$  δυαδικά ψηφία να παραστήσουμε προσημασμένους ακέραιους αριθμούς, τότε εκμεταλλευόμαστε το αριστερότερο bit (δηλαδή το MSB) του αριθμού, στο οποίο κωδικοποιούμε το πρόσημό του. Αν το πρόσημο έχει την τιμή 0, τότε ο αριθμός είναι θετικός, ενώ αν έχει την τιμή 1 είναι αρνητικός. Με τα υπόλοιπα  $N-1$  δυαδικά ψηφία κωδικοποιούμε την απόλυτη τιμή του αριθμού, δηλαδή το *μέτρο* του.



Οι αριθμοί  $01110010_{(2)}$  και  $00000001_{(2)}$  είναι θετικοί, ενώ οι αριθμοί  $11110010_{(2)}$  και  $11100001_{(2)}$  είναι αρνητικοί.

Ένα θετικό αριθμό τον παριστάνουμε θέτοντας το πιο σημαντικό bit (δηλαδή το πρόσημο) στην τιμή 0, και τα υπόλοιπα  $n-1$  bits στην τιμή του μέτρου του, δηλαδή στην τιμή του αριθμού. Επειδή ο μεγαλύτερος ακέραιος αριθμός που παριστάνεται με  $n-1$  bits είναι ο  $2^{n-1} - 1$ , η τιμή του αριθμού δεν μπορεί να ξεπερνά το όριο αυτό.

Θετικοί  
αριθμοί

Σε έναν υπολογιστή όπου το μήκος λέξης είναι 8, ο αριθμός 23 θα παρασταθεί ως  $00010111$ . Το αριστερότερο bit δηλώνει ότι ο αριθμός είναι θετικός, και τα υπόλοιπα bits περιέχουν τον αριθμό 23. Ο μεγαλύτερος θετικός αριθμός που μπορεί να αποθηκευθεί με 8 bits είναι ο  $01111111$ , δηλαδή ο  $127$  ( $2^{8-1} - 1 = 2^7 - 1 = 128 - 1$ ).

Υπάρχουν τρεις διαφορετικοί τρόποι για να κωδικοποιήσουμε τους αρνητικούς προσημασμένους αριθμούς στα υπόλοιπα  $n-1$  bits:

Αρνητικοί  
αριθμοί

- ♦ Η παράσταση **μέτρου**
- ♦ Η παράσταση **συμπληρώματος ως προς 1**
- ♦ Η παράσταση **συμπληρώματος ως προς 2**

Στη συνέχεια θα περιοριστούμε στην παράσταση συμπληρώματος ως προς 2.

Η παράσταση συμπληρώματος ως προς 2 είναι αυτή που χρησιμοποιείται περισσότερο, γιατί διευκολύνει και απλοποιεί πολύ την εκτέλεση των αριθμητικών πράξεων και για τους θετικούς και για τους αρνητικούς αριθμούς.

Εάν το πιο σημαντικό ψηφίο του αριθμού είναι 1, τότε ο αριθμός είναι αρνητικός. Για να βρούμε το μέτρο του αριθμού πρέπει να υπολογίσουμε το συμπλήρωμα ως προς 2 και των  $n$  ψηφίων του (δηλαδή λαμβάνουμε υπόψη και το πρόσημο). Το συμπλήρωμα ως προς 2 ενός δυαδικού αριθμού βρίσκεται, εάν αντικαταστήσουμε το 0 με 1 και το 1 με 0 και στη συνέχεια προσθέσουμε 1.

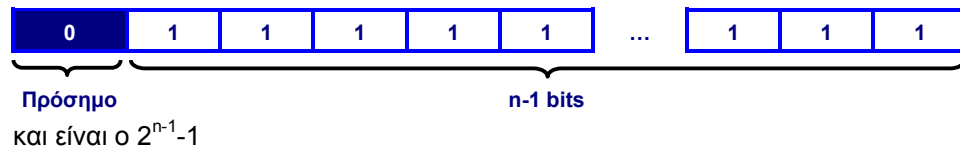
Για να μετατρέψουμε ένα αρνητικό στην παράσταση συμπληρώματος του 2, ακολουθούμε παρόμοια διαδικασία: γράφουμε το μέτρο του σε δυαδική μορφή, αντικαθιστούμε το 0 με 1 και το 1 με 0 και στη συνέχεια προσθέτουμε 1. Αν δεν

υπάρχει ήδη ως κρατούμενο, τοποθετούμε στα αριστερά του αριθμού το ψηφίο 1 του προσήμου.

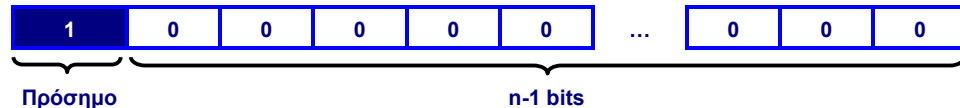
Για να βρούμε την παράσταση συμπληρώματος ως προς 2 του αριθμού -17 σε ένα υπολογιστή με μήκος λέξης 16 bits, αρχικά θα γράψουμε τον αντίστοιχο θετικό (17) σε δυαδική μορφή, δηλαδή 0000000000010001. Στη συνέχεια θα αντικαταστήσουμε το 0 με 1 και το 1 με 0 στον αριθμό αυτό, και θα πάρουμε 111111111101110. Στον αριθμό αυτό θα προσθέσουμε τον 1. Η τελική του παράσταση θα είναι λοιπόν 111111111101111.

Για να βρούμε την τιμή που παριστάνει ο αριθμός 11100110 θα υπολογίσουμε το συμπλήρωμά του ως προς 2 *μαζί με το πρόσημο*. Αρχικά αντιστρέφουμε όλα τα ψηφία του και παίρνουμε 00011001. Μετά προσθέτουμε τον αριθμό 1, και έχουμε  $00011001 + 1 = 00011010$ . Άρα το μέτρο του αριθμού είναι το  $00011010_{(2)} = 26_{(10)}$  και ο αριθμός είναι ο -26.

Ο μέγιστος θετικός αριθμός που μπορεί να παρασταθεί στο σύστημα αυτό με λέξη μήκους  $n$  bits έχει την παράσταση:



Ο ελάχιστος αρνητικός που μπορεί να παρασταθεί με μήκος λέξης  $n$  bits έχει την παράσταση:



Για να βρούμε την τιμή του αριθμού αυτού θα υπολογίσουμε το συμπλήρωμά του ως προς 2. Αντιστρέφουμε τα ψηφία του και παίρνουμε τον αριθμό  $01111...1111_{(2)}$  και μετά προσθέτουμε 1, για να πάρουμε τον  $10000...000_{(2)} = 2^{n-1}$ . Άρα ο ελάχιστος αριθμός είναι ο  $-2^{n-1}$ .

Η παράσταση συμπληρώματος του 2 έχει από ό,τι βλέπουμε μία ιδιαιτερότητα: ο μικρότερος αρνητικός αριθμός που μπορούμε να παραστήσουμε έχει μεγαλύτερη απόλυτη τιμή ( $2^{n-1}$ ) από το μεγαλύτερο θετικό ( $2^{n-1}-1$ ). Αυτό συμβαίνει γιατί ο αντίστοιχός του θετικός, ο  $2^{n-1}$ , χρειάζεται και το ψηφίο του προσήμου για να παρασταθεί.

Ας μην ξεχνάμε ότι οι έννοιες «συμπλήρωμα ως προς 2» και «παράσταση συμπληρώματος ως προς 2» είναι διαφορετικές. Το συμπλήρωμα ως προς 2 ενός αριθμού είναι το αποτέλεσμα της αντιστροφής των ψηφίων του αριθμού από 0 σε 1 και από 1 σε 0, και της πρόσθεσης σε αυτόν του 1. Η παράσταση συμπληρώματος ως προς δύο χρησιμοποιεί το συμπλήρωμα ως προς 2 για να παραστήσει τους αρνητικούς αριθμούς.

Όπως στο δυαδικό σύστημα υπάρχει η παράσταση συμπληρώματος ως προς 2, έτσι και στο δεκαδικό σύστημα υπάρχει και η παράσταση συμπληρώματος ως προς 10.

## Πράξεις προσημασμένων ακεραίων αριθμών

Η παράσταση του συμπληρώματος ως προς 2 έχει ένα πολύ μεγάλο πλεονέκτημα: οι πράξεις μεταξύ των αριθμών γίνονται απευθείας, χωρίς να χρειάζεται μετατροπή τους, ανεξάρτητα από το πρόσημό τους. Έτσι η διαδικασία της πρόσθεσης που είδαμε στην αρχή του μαθήματος εφαρμόζεται αυτούσια και σε προσημασμένους αριθμούς.

00010101	11001010	00010010	11111101
+	+	+	+
00110011	01000100	10001101	11110010
<hr/>			
01001000	100001110	10011111	111101111

Στο δεύτερο άθροισμα, το αποτέλεσμα της πρόσθεσης έχει και ένα επιπλέον bit, γιατί  $11001010_{(2)} + 01000100_{(2)} = 100001110_{(2)}$ . Αυτό το επιπλέον bit το αγνοούμε και παίρνουμε το σωστό αποτέλεσμα. Το ίδιο ισχύει και για το τέταρτο άθροισμα, αφού  $11111101_{(2)} + 11110010_{(2)} = 111101111_{(2)}$ .

Η μόνη περίπτωση στην εκτέλεση των πράξεων που χρειάζεται προσοχή είναι όταν το αποτέλεσμα μίας πράξης είναι πολύ μεγάλο ή πολύ μικρό και δεν μπορεί να παρασταθεί με το πλήθος των bits που έχουμε στη διάθεσή μας. Τότε λέμε ότι η πράξη προκάλεσε *υπερχείλιση* (overflow) του αποτελέσματος.

10010010	Το άθροισμα των αριθμών -110 και -33, που είναι -143, δεν μπορεί να παρασταθεί με 8 bits, γιατί ο μικρότερος αριθμός που μπορεί να παρασταθεί με 8 bits είναι ο -128. Αν προσθέσουμε αυτούς τους δύο αριθμούς, όπως βλέπουμε και στο σχήμα, το αποτέλεσμα είναι λανθασμένο.
11011111	
<hr/>	
01110001	

*αγνοείται*

Οι υπολογιστές διαθέτουν εσωτερικούς μηχανισμούς για να εντοπίζουν τις περιπτώσεις που μία αριθμητική πράξη προκαλεί υπερχείλιση του αποτελέσμά της.

## Παράσταση πραγματικών αριθμών

Εκτός από τους ακέραιους αριθμούς, θέλουμε να παραστήσουμε στον υπολογιστή και πραγματικούς αριθμούς, δηλαδή αριθμούς με ακέραιο και κλασματικό μέρος.

### Παράσταση σταθερής υποδιαστολής

Ένας απλός τρόπος για την παράσταση των πραγματικών αριθμών με λέξεις μήκους  $n$  bits είναι να μοιράσουμε τα ψηφία του αριθμού μεταξύ του ακέραιου και του κλασματικού μέρους, δηλαδή να παριστάνουμε το ακέραιο μέρος του αριθμού με  $n_1$  bits και το κλασματικό μέρος του με  $n_2$  ψηφία, με  $n_1 + n_2 = n$ . το πρόσημο του αριθμού θα κωδικοποιείται σαν και τα  $n$  ψηφία του αριθμού να παρίσταναν έναν ακέραιο, προσθέτοντας τη μονάδα στο δεξιότερο κλασματικό του ψηφίο. Αυτή είναι η παράσταση *σταθερής υποδιαστολής* (fixed point representation).

Για την παράσταση πραγματικών αριθμών διαθέτουμε 8 ψηφία, και διαθέτουμε 5 ψηφία στο ακέραιο μέρος και 3 ψηφία στο κλασματικό. Η παράσταση των αρνητικών αριθμών γίνεται με το συμπλήρωμα του 2. Ο μεγαλύτερος θετικός αριθμός που μπορούμε να παραστήσουμε έχει την

παράσταση 01111,111 και είναι ο 15,875. Ο μικρότερος αρνητικός αριθμός που μπορούμε να παραστήσουμε είναι ο 10000,000 που έχει την τιμή -16.

Οι πλησιέστεροι αριθμοί στο 0 που μπορούμε να παραστήσουμε (εκτός από το 0 βέβαια, που παριστάνεται ως 00000,000) είναι:

- 00000,001<sub>(2)</sub> = 0,125 για τους θετικούς και
- 11111,111<sub>(2)</sub> = -0,125 για τους αρνητικούς.

Στην παράσταση σταθερής υποδιαστολής οι πράξεις γίνονται ακριβώς όπως και στους ακέραιους αριθμούς.

Το άθροισμα των αριθμών 01001,110<sub>(2)</sub> = 9,75 και 10010,001<sub>(2)</sub> = -13,875 υπολογίζεται απευθείας και είναι 11011,111<sub>(2)</sub> = -4,125

Το σπουδαιότερο μειονέκτημα της παραστάσεως σταθερής υποδιαστολής είναι ότι το διάστημα των αριθμών που μπορούν να παρασταθούν δεν είναι πολύ μεγάλο.

### Παράσταση κινητής υποδιαστολής

Συνήθως στους υπολογιστές χρησιμοποιείται η παράσταση *κινητής υποδιαστολής* (floating point representation).

$$N = \sigma \cdot 2^{\epsilon}$$

**Συντελεστής (mantissa).**  
Έχει το ίδιο πρόσημο με τον αριθμό N.

**Εκθέτης (exponent).**

Στην παράσταση κινητής υποδιαστολής ο δυαδικός αριθμός N που θέλουμε να παραστήσουμε εκφράζεται σε *εκθετική μορφή* (exponential representation), σαν ένα γινόμενο δηλαδή ενός κλασματικού αριθμού και μιας δύναμης του 2.

Ο αριθμός 101,011<sub>(2)</sub> σε εκθετική μορφή μπορεί να γραφτεί με διάφορες μορφές:

- $0,101011 \cdot 2^3 \Leftrightarrow \sigma = 0,101011, \quad \epsilon = 3_{(10)} = 11_{(2)}$
- $1,01011 \cdot 2^2 \Leftrightarrow \sigma = 1,01011, \quad \epsilon = 2_{(10)} = 10_{(2)}$
- $10,1011 \cdot 2^1 \Leftrightarrow \sigma = 10,1011, \quad \epsilon = 1_{(10)} = 1_{(2)}$

Το ίδιο συμβαίνει και στο δεκαδικό σύστημα: κάθε αριθμός μπορεί να γραφτεί με πολλές εκθετικές μορφές. Το 1023 π.χ. μπορεί να γραφτεί σαν  $1,023 \cdot 10^3$ , σαν  $10,23 \cdot 10^2$ , σαν  $0,001023 \cdot 10^6$  κλπ.

Βλέπουμε λοιπόν ότι υπάρχουν πολλές εναλλακτικές παραστάσεις ενός αριθμού σε εκθετική μορφή. Στους υπολογιστές έχει επιλεγεί μία από τις παραστάσεις αυτές, η οποία έχει την ιδιότητα  $\frac{1}{2} \leq \sigma < 1$  και ονομάζεται *κανονική μορφή* (normal form).

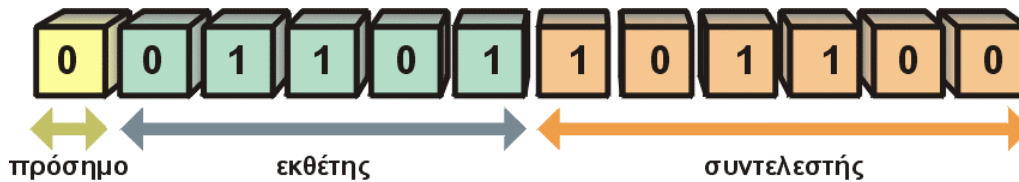
Όταν ο συντελεστής είναι μεταξύ  $\frac{1}{2}$  και 1, έχει δύο χαρακτηριστικά:

- Το **ακέραιο μέρος του είναι πάντα 0**. Έτσι δε χρειάζεται να το αποθηκεύουμε, γιατί η τιμή του είναι γνωστή και δεδομένη.
- Το **πρώτο του κλασματικό ψηφίο είναι πάντα 1**. Αυτό συμβαίνει, γιατί, οι κλασματικοί αριθμοί που είναι μεγαλύτεροι ή ίσοι από  $\frac{1}{2}$  ( $=2^{-1}$ ), στο δυαδικό σύστημα περιέχουν πάντα τον προσθετέο  $2^{-1}$ .

Από όλες τις εκθετικές μορφές του αριθμού  $101,011_{(2)}$  που είδαμε πιο πριν, η κανονική μορφή είναι η  $0,101011 \cdot 2^3$ .

Η κανονική μορφή του αριθμού  $0,000100_{(2)}$  είναι η  $0,100 \cdot 2^{-3}$ . Εδώ ο εκθέτης πρέπει να είναι αρνητικός, για να ικανοποιεί ο συντελεστής τη συνθήκη  $\frac{1}{2} \leq \sigma < 1$ .

Αν λοιπόν όλοι οι αριθμοί είναι εκφρασμένοι στην κανονική εκθετική μορφή, μπορούμε να τους κωδικοποιήσουμε αφιερώνοντας  $n_1$  δυαδικά ψηφία στον εκθέτη και  $n_2$  δυαδικά ψηφία στο συντελεστή, κρατώντας και ένα ψηφίο που θα κωδικοποιεί το πρόσημο του αριθμού, όπως βλέπουμε στο σχήμα.



Το κλασματικό μέρος του συντελεστή παριστάνεται σαν ένας δυαδικός αριθμός με  $n_1$  ψηφία. Εάν ο συντελεστής έχει λιγότερα από  $n_1$  ψηφία, προσθέτουμε μηδενικά στο τέλος, ενώ αν έχει περισσότερα από  $n_1$  ψηφία, τότε τον στρογγυλοποιούμε. Στη *στρογγυλοποίηση* (rounding), αγνοούμε τα ψηφία που περισσεύουν, αλλά, εάν το πρώτο ψηφίο που περισσεύει είναι 1, τότε προσθέτουμε 1 στο λιγότερο σημαντικό ψηφίο του συντελεστή. Ο εκθέτης παριστάνεται και αυτός σαν ένας δυαδικός αριθμός με  $n_2$  ψηφία.

Το πιο σημαντικό από τα ψηφία της λέξης, που παίζει το ρόλο του πρόσημου: έχει την τιμή 0, αν ο αριθμός είναι θετικός και την τιμή 1 αν είναι αρνητικός. Αυτό είναι το πρόσημο του αριθμού· ο εκθέτης, όπως είδαμε, μπορεί να είναι αρνητικός, οπότε έχει το δικό του πρόσημο.

Οι πράξεις με πραγματικούς αριθμούς κινητής υποδιαστολής είναι πιο πολύπλοκες από ό,τι με τους ακεραίους.

Για να προσθέσουμε δύο πραγματικούς αριθμούς κινητής υποδιαστολής, πρέπει πρώτα να τους μετατρέψουμε ώστε να έχουν τον ίδιο εκθέτη. Αν ο ένας αριθμός έχει εκθέτη  $e_1$  και ο άλλος  $e_2$ , και ισχύει  $e_1 < e_2$ , τότε αυξάνουμε τον  $e_1$  κατά  $e_2 - e_1$  και «ολισθαίνουμε» το συντελεστή του αριθμού αυτού προς τα δεξιά κατά  $e_2 - e_1$  ψηφία. Κατά την ολίσθηση αυτή, τα δεξιότερα ψηφία του αριθμού χάνονται, έτσι ο αριθμός μπορεί να μεταβληθεί. Το τελικό αποτέλεσμα λοιπόν μπορεί να μην είναι ακριβές.

Στη συνέχεια προσθέτουμε τους συντελεστές των αριθμών και γράφουμε ξανά το αποτέλεσμα στην κανονική μορφή στρογγυλοποιώντας το συντελεστή. Σε όλες τις μετατροπές όμως το πλήθος των ψηφίων του συντελεστή παραμένει σταθερό.

Στην παράσταση κινητής υποδιαστολής με 8 ψηφία για το συντελεστή και 4 ψηφία για τον εκθέτη, ο αριθμός  $x = 16,125_{(10)}$  παριστάνεται ως  $0,10000001 \cdot 2^5$  και ο αριθμός  $y = 4,3125_{(10)}$  παριστάνεται ως  $0,1000101 \cdot 2^3$ .

Πρώτα μετατρέπουμε τον αριθμό με το μικρότερο εκθέτη, που είναι ο  $y$ . Αυξάνουμε τον εκθέτη του κατά 2 και ολισθαίνουμε το συντελεστή του προς τα δεξιά κατά 2 ψηφία.

$x = 0010110000001$   
 $y = 0001110001010$

$y = 0010100100010$

$$x+y = 0010110100011$$

Στη συνέχεια προσθέτουμε τους δύο συντελεστές. Το άθροισμα δε χρειάζεται κανονικοποίηση, άρα είναι και το τελικό αποτέλεσμα. Η τιμή του είναι  $0,10100011 \cdot 2^5$ , δηλαδή 20,375. Το σωστό αποτέλεσμα της πρόσθεσης όμως είναι 20,4375. Το σφάλμα οφείλεται στη μετατροπή του  $y$  ώστε να έχει τον ίδιο εκθέτη με το  $x$ .

Οι πράξεις του πολλαπλασιασμού και της διαίρεσης με πραγματικούς αριθμούς κινητής υποδιαστολής είναι πιο εύκολες.

$$\begin{aligned} (\sigma_1 \cdot 2^{\varepsilon_1}) \cdot (\sigma_2 \cdot 2^{\varepsilon_2}) &= (\sigma_1 \cdot \sigma_2) \cdot 2^{\varepsilon_1 + \varepsilon_2} \\ (\sigma_1 \cdot 2^{\varepsilon_1}) / (\sigma_2 \cdot 2^{\varepsilon_2}) &= (\sigma_1 / \sigma_2) \cdot 2^{\varepsilon_1 - \varepsilon_2} \end{aligned}$$

Για να πολλαπλασιάσουμε δύο αριθμούς προσθέτουμε τους εκθέτες τους και πολλαπλασιάζουμε τους συντελεστές· μετά φέρνουμε πάλι το αποτέλεσμα στην κανονική μορφή. Για να διαιρέσουμε δύο αριθμούς αφαιρούμε τους εκθέτες, διαιρούμε τους συντελεστές και κανονικοποιούμε το αποτέλεσμα.

Όπως είδαμε και στο προηγούμενο παράδειγμα, πολλές φορές η μετατροπή ή η στρογγυλοποίηση που κάνουμε στους αριθμούς κατά την εκτέλεση των πράξεων επιφέρει ένα μικρό σφάλμα στο αποτέλεσμα. Μετά από μία μεγάλη σειρά πράξεων λοιπόν, τα σφάλματα αυτά «συσσωρεύονται», οπότε τα αποτελέσματα μπορεί να μην είναι ικανοποιητικά.

Το πλήθος  $n_1$  των ψηφίων που χρησιμοποιούνται για τον συντελεστή καθορίζουν αντιστοίχως την ακρίβεια παραστάσεως των αριθμών και το πλήθος  $n_2$  των ψηφίων που χρησιμοποιούνται για τον εκθέτη καθορίζουν το εύρος των αριθμών που μπορούμε να παραστήσουμε.

Στην παράσταση κινητής υποδιαστολής με 5 ψηφία για το συντελεστή και 3 ψηφία για τον εκθέτη, οι κοντινότεροι αριθμοί στο 0 που μπορούμε να παραστήσουμε είναι οι:

$$000000001 = 0,03125_{(10)} \text{ (θετικός)}$$

$$111111111 = -0,03125_{(10)} \text{ (αρνητικός)}$$

Οι αριθμοί αυτοί καθορίζουν την ακρίβεια της παράστασης, γιατί οποιοσδήποτε μικρότερος κλασματικός αριθμός θα παρασταθεί σαν 0.

Ο μεγαλύτερος και ο μικρότερος αριθμός που μπορούν να παρασταθούν είναι οι:

$$011111111 = 0,96875 \cdot 2^7 = 124_{(10)}$$

$$100000001 = -0,96875 \cdot 2^7 = -124_{(10)}$$

Οι αριθμοί αυτοί καθορίζουν το εύρος της αναπαράστασης.

Βλέπουμε ότι όταν ο εκθέτης ενός αριθμού είναι μεγάλος, τα κλασματικά ψηφία της τιμής του είναι λίγα (π.χ. ο μεγαλύτερος αριθμός εδώ δεν έχει κανένα κλασματικό ψηφίο). Αντίστροφα, όταν ο εκθέτης είναι μικρός, η τιμή του αριθμού έχει πιο πολλά κλασματικά ψηφία, αλλά είναι πιο μικρή.



## Παράσταση χαρακτήρων

Εκτός από αριθμούς, θέλουμε να παραστήσουμε στον υπολογιστή και σύμβολα, όπως είναι τα γράμματα, τα σημεία στίξης, τα αριθμητικά ψηφία. Αυτά τα σύμβολα ονομάζονται *χαρακτήρες* (characters). Όπως οι αριθμοί και όλες οι άλλες πληροφορίες, έτσι και οι χαρακτήρες παριστάνονται στους υπολογιστές σαν μία σειρά από δυαδικά ψηφία.

Κάθε χαρακτήρας αντιστοιχίζεται σε μία σειρά ψηφίων που έχει μήκος συνήθως 8 ή 16 bits. Ένα κείμενο παριστάνεται με την ακολουθία των χαρακτήρων που το αποτελούν.

Η παράσταση των αριθμών με δυαδικά ψηφία είναι πολύ απλή και «αυτονόητη». Με τους χαρακτήρες όμως δε συμβαίνει το ίδιο. Η αντιστοίχιση χαρακτήρων με ακολουθίες από δυαδικά ψηφία μπορεί να γίνει με πάρα πολλούς τρόπους· έτσι για να μπορούν οι υπολογιστές να χρησιμοποιούν από κοινού δεδομένα πρέπει να συμφωνούν στην αντιστοίχιση αυτή.

Αν ένας υπολογιστής  $Y_1$  παριστάνει το γράμμα «Α» με την ακολουθία των δυαδικών ψηφίων 00110100, και ένας υπολογιστής  $Y_2$  παριστάνει με την ακολουθία αυτή το γράμμα «Ε», τότε όταν ο  $Y_1$  στείλει ένα κείμενο στον  $Y_2$ , όλα τα «Α» στο κείμενο θα ερμηνεύονται σαν «Ε»!

Μία κοινώς αποδεκτή αντιστοίχιση χαρακτήρων-δυαδικών ακολουθιών ονομάζεται *σύνολο χαρακτήρων* (character set) και συνήθως είναι καθορισμένη από κάποιο διεθνή οργανισμό προτυποποίησης.

Για ευκολία, επειδή μία ακολουθία δυαδικών ψηφίων ερμηνεύεται πολύ εύκολα σαν ένας δυαδικός αριθμός, το σύνολο χαρακτήρων αναφέρεται σαν αντιστοιχία χαρακτήρων με αριθμούς.

Αν το γράμμα «Α» παριστάνεται με τη δυαδική ακολουθία 01000001, που σαν δυαδικός αριθμός έχει την τιμή 65, λέμε ότι το γράμμα «Α» παριστάνεται με τον αριθμό 65.

Ένα από τα πρώτα σύνολα χαρακτήρων είναι το ASCII (American Standard Code for Information Interchange), το οποίο στην αρχή παρίστανε τους χαρακτήρες με 7 bits και περιλάμβανε μόνο το λατινικό αλφάβητο. Το 8<sup>ο</sup> bit χρησιμοποιούνταν ως *bit ισοτιμίας* (parity bit) που δήλωνε το πλήθος των bits του με την τιμή 1: αν το πλήθος ήταν περιττό, τότε το 8<sup>ο</sup> bit ήταν 1, αλλιώς ήταν 0. Το bit ισοτιμίας χρησίμευε στην ανίχνευση λαθών κατά τη μετάδοση δεδομένων.

Ο χαρακτήρας 0110101<sub>(2)</sub> έχει άρτιο πλήθος bits με την τιμή 1, έτσι το bit ισοτιμίας του είναι 0, και η τελική του μορφή είναι **0110101**.

Ο χαρακτήρας 1001111<sub>(2)</sub> έχει περιττό πλήθος bits με την τιμή 1, έτσι το bit ισοτιμίας του είναι 1, και η τελική του μορφή είναι **11001111**.

Αργότερα το πρότυπο ASCII επεκτάθηκε στα 8 bits για να περιλάβει και άλλους χαρακτήρες (π.χ. χαρακτήρες με διακριτικά).

Οι πρώτοι 32 χαρακτήρες του προτύπου ASCII, με τιμές 0-31 είναι «μη εκτυπώσιμοι» (non printable) και χρησιμοποιούνται σαν μηνύματα ελέγχου για οθόνες, εκτυπωτές κλπ.

Ο χαρακτήρας 10 δηλώνει την αλλαγή γραμμής σε μία οθόνη τερματικού, ενώ ο χαρακτήρας 12 δηλώνει το τέλος σελίδας σε μία εκτύπωση.

Για την Ελληνική γλώσσα χρησιμοποιούμε συνήθως μία παραλλαγή του ASCII που έχει μήκος 8 bits και έχει προτυποποιηθεί από τον Ελληνικό Οργανισμό Τυποποίησης με το όνομα ΕΛΟΤ-928. Τα γράμματα του Ελληνικού αλφαβήτου παριστάνονται με τιμές που έχουν 1 στο αριστερότερο bit (δηλαδή τιμές μεγαλύτερες από 127).

Ας δούμε τις πιο ενδιαφέρουσες αντιστοιχίες χαρακτήρων-αριθμών από το πρότυπο ΕΛΟΤ-928:

33	!	64	@	95	_	126	~	192	ĩ	224	ü
34	"	65	A	96	`	161	˘	193	À	225	α
35	#	66	B	97	a	162	Á	194	Β	226	β
36	\$	67	C	98	b	163	£	195	Γ	227	γ
37	%	68	D	99	c	164	¤	196	Δ	228	δ
38	&	69	E	100	d	165	¥	197	Ε	229	ε
39	'	70	F	101	e	166	ı	198	Ζ	230	ζ
40	(	71	G	102	f	167	§	199	Η	231	η
41	)	72	H	103	g	168	¨	200	Θ	232	θ
42	*	73	I	104	h	169	©	201	Ι	233	ι
43	+	74	J	105	i	171	«	202	Κ	234	κ
44	,	75	K	106	j	172	¬	203	Λ	235	λ
45	-	76	L	107	k	173	-	204	Μ	236	μ
46	.	77	M	108	l	174	®	205	Ν	237	ν
47	/	78	N	109	m	175	—	206	Ξ	238	ξ
48	0	79	O	110	n	176	°	207	Ο	239	ο
49	1	80	P	111	o	177	±	208	Π	240	π
50	2	81	Q	112	p	178	²	209	Ρ	241	ρ
51	3	82	R	113	q	179	³	211	Σ	242	ς
52	4	83	S	114	r	180	´	212	Τ	243	σ
53	5	84	T	115	s	181	μ	213	Υ	244	τ
54	6	85	U	116	t	182	¶	214	Φ	245	υ
55	7	86	V	117	u	183	·	215	Χ	246	φ
56	8	87	W	118	v	184	Έ	216	Ψ	247	χ
57	9	88	X	119	w	185	Ή	217	Ω	248	ψ
58	:	89	Y	120	x	186	ΐ	218	Ϊ	249	ω
59	;	90	Z	121	y	187	»	219	Ϋ	250	ϊ
60	<	91	[	122	z	188	Ό	220	ά	251	ü
61	=	92	\	123	{	189	½	221	έ	252	ό
62	>	93	]	124		190	Ύ	222	ή	253	ύ
63	?	94	^	125	}	191	Ώ	223	ί	254	ώ

### Το πρότυπο Unicode

Επειδή οι 256 διαφορετικοί χαρακτήρες που διαθέτει το πρότυπο ASCII δεν επαρκούν για να κωδικοποιηθούν όλα τα αλφάβητα, ιδιαίτερα εκείνα που έχουν ιδεογράμματα, έχει αναπτυχθεί και το πρότυπο Unicode, το οποίο παριστάνει ένα χαρακτήρα με μία ακολουθία 16 bits. Έτσι μπορεί να παραστήσει 65536 χαρακτήρες, που είναι αρκετοί για γλώσσες όπως η Κινέζικη και η Ιαπωνική.

Οι 65536 χαρακτήρες του προτύπου Unicode έχουν διαιρεθεί σε «περιοχές», κάθε μία από τις οποίες περιέχει την κωδικοποίηση των χαρακτήρων μίας γλώσσας. Οι πρώτοι 256 χαρακτήρες, για λόγους συμβατότητας, έχουν αποδοθεί στους ίδιους χαρακτήρες με το πρότυπο ASCII.

Οι ελληνικοί χαρακτήρες στο πρότυπο Unicode βρίσκονται στην περιοχή από 880<sub>(10)</sub> έως 975<sub>(10)</sub>.

Οι χαρακτήρες της Ταϊλανδέζικης γραφής βρίσκονται στην περιοχή από 3584<sub>(10)</sub> έως 3711<sub>(10)</sub>.

Το μεγάλο πλεονέκτημα του προτύπου Unicode, εκτός τού ότι μπορεί να στεγάσει τους χαρακτήρες όλων των συστημάτων γραφής χωρίς υπερκαλύψεις, είναι ότι μπορεί να χρησιμοποιηθεί για να κωδικοποιήσουμε ένα κείμενο που περιέχει πολλές διαφορετικές γλώσσες.

Το τρίγωνο κείμενο της συμφωνίας εμπορικής συνεργασίας μεταξύ Ελλάδας, Αρμενίας και Γεωργίας περιέχει τρία διαφορετικά αλφάβητα (τέσσερα αν περιέχει λέξεις ή φράσεις που είναι γραμμένες με το λατινικό αλφάβητο).

Αν το κείμενο παρασταθεί με το πρότυπο Unicode, όλα τα κείμενα θα είναι κωδικοποιημένα με τους σωστούς χαρακτήρες. Αν πρέπει να χρησιμοποιηθεί το πρότυπο ASCII, θα πρέπει να γραφτούν τρία διαφορετικά κείμενα και κάθε ένα να «διαβάζεται» με την έκδοση του προτύπου ASCII που περιλαμβάνει την αντίστοιχη γλώσσα.

Οι χαρακτήρες που θα περιέχονται σε κάθε κείμενο θα έχουν κωδικοποιήσεις με τιμές (που φαίνονται στον πίνακα:

Γλώσσα	Από	Έως
Ελληνική	880 <sub>(10)</sub>	975 <sub>(10)</sub>
Αρμενική	1328 <sub>(10)</sub>	1423 <sub>(10)</sub>
Γεωργιανή	4256 <sub>(10)</sub>	4351 <sub>(10)</sub>



Οι πράξεις μεταξύ δυαδικών ακεραίων αριθμών γίνονται παρόμοια με τις πράξεις στο δεκαδικό σύστημα. Ο πιο συνηθισμένος τρόπος παράστασης των αρνητικών ακεραίων αριθμών είναι η παράσταση συμπληρώματος του 2, η οποία διευκολύνει πολύ στην εκτέλεση των αριθμητικών πράξεων. Για την παράσταση των πραγματικών αριθμών υπάρχει η παράσταση σταθερής υποδιαστολής, όπου κάθε αριθμός έχει δεδομένο πλήθος κλασματικών ψηφίων, και η παράσταση κινητής υποδιαστολής, όπου οι αριθμοί παριστάνονται σε κανονική εκθετική μορφή. Η δεύτερη είναι πιο ευέλικτη, αν και οι αριθμητικές πράξεις σε αυτή είναι πιο πολύπλοκες, γι' αυτό και γενικά προτιμάται. Οι χαρακτήρες παριστάνονται σαν ακολουθίες από 8 ή 16 δυαδικά ψηφία, σύμφωνα με ένα πρότυπο κωδικοποίησης όπως το ASCII.



Bit ισοτιμίας	Parity Bit
Εκθέτης	Exponent
Εκθετική Μορφή	Exponential Representation
Κανονική Μορφή	Normal Form
Λέξη	Word
Μήκος Λέξης	Word Length
Παράσταση Σταθερής Υποδιαστολής	Fixed Point Representation
Παράσταση Κινητής Υποδιαστολής	Floating Point Representation
Σύνολο Χαρακτήρων	Character Set
Συντελεστής	Mantissa
Στρογγυλοποίηση	Rounding
Υπερχείλιση	Overflow
Χαρακτήρας	Character

## Ερωτήσεις

- ? Ποιο πρέπει να είναι το άθροισμα τριών δυαδικών ψηφίων για να έχουμε κρατούμενο;
- ? Γιατί η παράσταση συμπληρώματος του 2 είναι τόσο δημοφιλής;
- ? Πώς παριστάνεται ο ακεραίος αριθμός -17 σε έναν υπολογιστή με μήκος λέξης 16 bits και παράσταση συμπληρώματος του 2 για αρνητικούς αριθμούς;
- ? Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα για την παράσταση σταθερής υποδιαστολής των πραγματικών αριθμών;
- ? Πώς προσθέτουμε δύο αριθμούς σε παράσταση κινητής υποδιαστολής;
- ? Πώς σχετίζεται η ακρίβεια και το εύρος των αριθμών κινητής υποδιαστολής;
- ? Τι εννοούμε με τους όρους «συμπλήρωμα ως προς 2» και «παράσταση συμπληρώματος ως προς 2»;
- ? Τι εννοούμε με τον όρο «υπερχείλιση»;

## Μάθημα 2.3

# Μέθοδοι Συμπίεσης Δεδομένων

Σκοπός του μαθήματος είναι να περιγράψει τις βασικές τεχνικές συμπίεσης που χρησιμοποιούνται για την μείωση του όγκου των ψηφιακών δεδομένων.

Σκοπός του μαθήματος

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- ♦ Να περιγράφεις τη συμπίεση δεδομένων
- ♦ Να απαριθμείς τις κατηγορίες συμπίεσης
- ♦ Να απαριθμείς τους γνωστότερους αλγορίθμους συμπίεσης και να περιγράφεις το πώς λειτουργούν

Τι θα μάθεις;

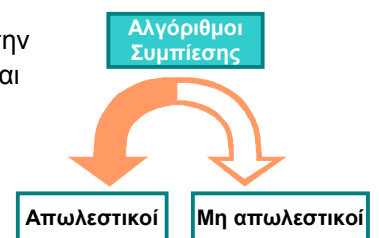
**Συμπίεση** (compression) μιας ακολουθίας δεδομένων, ονομάζουμε την ελάττωση του μεγέθους της ακολουθίας, ώστε να χρειάζεται λιγότερος χώρος για την αποθήκευση ή τη μετάδοσή της. Η διαδικασία της συμπίεσης εφαρμόζεται συστηματικά στα υπολογιστικά συστήματα που χρησιμοποιούν και επεξεργάζονται μεγάλο όγκο ψηφιακών δεδομένων.

Ένας αλγόριθμος συμπίεσης μπορεί να επιφέρει μεγάλη ελάττωση του μήκους σε ακολουθίες δεδομένων με κάποιο ιδιαίτερο χαρακτηριστικό, ενώ ο ίδιος να είναι αναποτελεσματικός για άλλες ακολουθίες, αφήνοντάς τις στο ίδιο μήκος ή ακόμα και μεγαλώνοντάς το σε μερικές περιπτώσεις.

Όταν έχουμε την ακολουθία των δεδομένων σε συμπιεσμένη μορφή, πρέπει να εφαρμοστεί η αντίστροφη διαδικασία της **αποσυμπίεσης** (decompression, extraction) προκειμένου τα δεδομένα να μπορούν να χρησιμοποιηθούν και πάλι. Η διαδικασία αυτή της αποσυμπίεσης των δεδομένων καθορίζει και τις κατηγορίες των μεθόδων συμπίεσης. Έτσι, υπάρχουν δύο κατηγορίες αλγορίθμων συμπίεσης, οι **απωλεστικοί** (lossy) και οι **μη απωλεστικοί** (lossless) αλγόριθμοι.

Στους απωλεστικούς αλγορίθμους, όταν γίνει η συμπίεση και μετά ακολουθήσει αποσυμπίεση των δεδομένων, η τελική ακολουθία των δεδομένων διαφέρει από την αρχική. Αντίθετα στους μη απωλεστικούς αλγορίθμους, η διαδικασία συμπίεσης και αποσυμπίεσης επαναφέρει την αρχική ακολουθία. Αν πρέπει να μεταφερθούν δεδομένα με απόλυτη ακρίβεια, πιστότητα, χωρίς να αλλοιωθεί το περιεχόμενό τους, πρέπει να εφαρμοστεί μια μη απωλεστική μέθοδος συμπίεσης. Εφαρμογές ή συστήματα που μεταδίδουν αναλλοίωτες πληροφορίες από το ένα μέσο στο άλλο, για παράδειγμα κάρτες δικτύου ή modem, χρησιμοποιούν τεχνικές μη απωλεστικής συμπίεσης. Υπάρχουν όμως εφαρμογές όπου η μικρή διαφοροποίηση από την αρχική μορφή των δεδομένων δεν επιφέρει σημαντικές αλλαγές. Έτσι, οι περισσότερες εφαρμογές που έχουν να κάνουν με σύνθετες μορφές δεδομένων όπως είναι ο ήχος, η εικόνα, το video, όπου το τελικό αποτέλεσμα αξιολογείται από τον

Είναι πολύ σημαντικό να γνωρίζουμε ότι η αποτελεσματικότητα μιας μεθόδου ή αλλιώς **αλγορίθμου συμπίεσης** (compression algorithm) εξαρτάται από τη μορφή των δεδομένων στα οποία εφαρμόζεται.

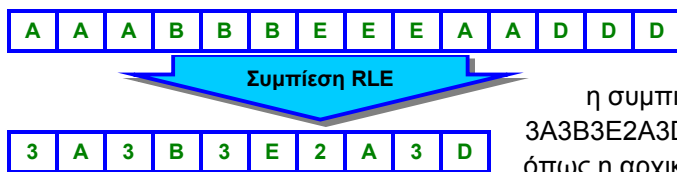


ανθρώπινο παράγοντα (αυτί, μάτι), μπορούν να κάνουν απωλεστική συμπίεση χωρίς πολλές φορές να υπάρχουν εμφανείς αλλοιώσεις στην ποιότητα των δεδομένων.

Στη συνέχεια θα εξετάσουμε τις αντιπροσωπευτικότερες τεχνικές μη απωλεστικής συμπίεσης.

### Αλγόριθμος RLE

Ο αλγόριθμος RLE (Run Length Encoding) αποτελεί μια από τις απλούστερες τεχνικές συμπίεσης. Σύμφωνα με τη μέθοδο αυτή, διατρέχεται η ακολουθία των bytes που αποτελούν τα δεδομένα προς συμπίεση και εντοπίζονται οι διαδοχικές επαναλήψεις του ίδιου χαρακτήρα. Στη συνέχεια αντικαθίστανται οι συνεχόμενες επαναλήψεις με το πλήθος τους, ακολουθούμενο από τον χαρακτήρα.



Αν συμπίεσουμε την ακολουθία AAABBBEEEEAAADD με τη μέθοδο RLE, τότε η συμπίεσμένη μορφή της ακολουθίας θα είναι 3A3B3E2A3D, που έχει μήκος 10 χαρακτήρες και όχι 14 όπως η αρχική.

Αν προσέξουμε το σχήμα της παραπάνω συμπίεσης θα παρατηρήσουμε ότι αποδίδει ικανοποιητικά για ακολουθίες δεδομένων που έχουν συχνές επαναλήψεις χαρακτήρων. Αν για παράδειγμα συμπίεσουμε την ακολουθία «ABCDE» μήκους 5 bytes με τον αλγόριθμο RLE, η συμπίεσμένη μορφή της είναι «1A1B1C1D1E», με μήκος 10. Στην περίπτωση αυτή όχι μόνο δεν έγινε συμπίεση, αλλά διπλασιάστηκε το μήκος της ακολουθίας. Για να αποφευχθούν αυτές οι περιπτώσεις, ο RLE δε συμπίεζει τις ακολουθίες μη συνεχόμενων χαρακτήρων.

Στην πράξη ο αλγόριθμος διατρέχει την ακολουθία των δεδομένων και αναζητά διαδοχικές επαναλήψεις του ίδιου χαρακτήρα. Όταν βρεθεί μεμονωμένος χαρακτήρας που δεν επαναλαμβάνεται, ή διπλή επανάληψη του, μένει ως έχει, καθώς δεν επωφελούμαστε από την κωδικοποίηση RLE που απαιτεί 2 χαρακτήρες. Όταν όμως βρεθούν διαδοχικές επαναλήψεις ενός χαρακτήρα, τις αντικαθιστά με ένα ζεύγος (αριθμός, χαρακτήρας), όπου «αριθμός» είναι το σύνολο των συνεχόμενων εμφανίσεων του χαρακτήρα.

### Αλγόριθμος του Huffman

Αυτή η τεχνική συμπίεσης παρουσιάστηκε από τον D. Huffman το 1952 και επιτυγχάνει μεγάλα ποσοστά συμπίεσης εκμεταλλευόμενη τη στατιστική ανάλυση της ακολουθίας δεδομένων. Η ανάλυση αυτή είναι πιο σύνθετη από την απλή ανίχνευση των συνεχόμενων επαναλήψεων που κάνει ο RLE.

Σύμφωνα με την κωδικοποίηση του προτύπου ASCII, σε κάθε χαρακτήρα αντιστοιχεί ένα byte, που είναι ο κωδικός ASCII του χαρακτήρα αυτού. Αν η κωδικοποίηση γίνεται με τον νέο κώδικα Unicode, χρειάζονται 2 bytes για κάθε χαρακτήρα. Το συνολικό μέγεθος της ακολουθίας σε bytes είναι ίσο με το πλήθος των χαρακτήρων επί 1 ή 2 bytes ανά χαρακτήρα (ανάλογα με την κωδικοποίηση).

Στην κωδικοποίηση Huffman, επιτυγχάνεται συμπίεση καθώς ο κώδικας που αντιστοιχεί σε κάθε χαρακτήρα δεν έχει σταθερό μήκος αλλά μεταβλητό. Η κωδικοποίηση για τους πιο συχνά εμφανιζόμενους χαρακτήρες είναι μικρότερη από ό,τι για τους λιγότερο συχνά εμφανιζόμενους, με αποτέλεσμα το συνολικό μέγεθος να είναι μικρότερο, καθώς για τους συχνότερους χρησιμοποιούνται μόνο 2-3 bits και όχι 8 ή 16.



Μια απλοποιημένη έκδοση του αλγορίθμου Huffman αποτελείται από τα εξής στάδια:

- Βήμα 1** Μετράμε τη συχνότητα του κάθε χαρακτήρα στην ακολουθία
- Βήμα 2:** Ταξινομούμε τις συχνότητες εμφάνισης σε μια λίστα κατά φθίνουσα τάξη
- Βήμα 3:** Κατασκευάζουμε ένα «δένδρο» για την κωδικοποίηση ξεκινώντας με τους συχνότερα εμφανιζόμενους χαρακτήρες
- Βήμα 4:** Αντιστοιχίζουμε τα δυαδικά '0' και '1' σε κάθε κόμβο του δέντρου: Αρχίζοντας από την ρίζα του δέντρου, προσθέτουμε '0' για κάθε αριστερό παιδί και '1' για κάθε δεξί. Οι χαρακτήρες που κωδικοποιούνται είναι τα φύλλα στην βάση του δέντρου. Αρχίζοντας από την κορυφή (ρίζα) του δέντρου, διατρέχοντας το μοναδικό μονοπάτι προς κάθε φύλλο, συλλέγουμε 0 ή 1 και ορίζουμε τον κώδικα για το χαρακτήρα που αντιστοιχεί στο φύλλο αυτό.

Έστω ότι θέλουμε να κωδικοποιήσουμε τη λέξη «ΑΛΛΟΣ».

**Βήμα 1,2:** Δημιουργούμε μια λίστα με τους χαρακτήρες ταξινομώντας τους με σειρά εμφάνισης. Έτσι έχουμε τη λίστα {«Λ», «Α», «Ο», «Σ»} αφού το «Λ» εμφανίζεται δύο φορές και οι υπόλοιποι χαρακτήρες από μία (τους οποίους και τοποθετούμε στη λίστα με τη σειρά εμφάνισης τους).

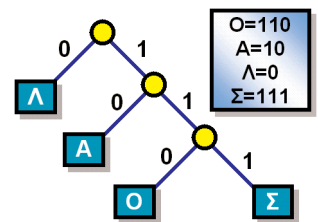
**Βήμα 3:** Ξεκινώντας από τη ρίζα, δημιουργούμε δύο παιδιά: το «Λ» (το γράμμα που εμφανίζεται πρώτο στη λίστα) και έναν εσωτερικό κόμβο. Στη συνέχεια από τον κόμβο αυτό δημιουργούμε πάλι δύο παιδιά, ένα με τον επόμενο χαρακτήρα στη λίστα, δηλαδή το «Α» και έναν εσωτερικό κόμβο. Διαγράφουμε το «Λ» από τη λίστα. Επαναλαμβάνουμε την διαδικασία μέχρι να τελειώσουν όλοι οι χαρακτήρες της λίστας.

Τελικά όλοι οι χαρακτήρες «Λ», «Α», «Ο», και «Σ» αποθηκεύονται σε φύλλα του δέντρου.

**Βήμα 4:** Αντιστοιχίζουμε τα δυαδικά 0 και 1, διατρέχοντας το μοναδικό μονοπάτι προς κάθε φύλλο-χαρακτήρα.

Τελικά, η ακολουθία «ΑΛΛΟΣ» που είχε αρχικό μήκος  $5 \times 8 = 40$  bits, συμπιεσμένη γίνεται: **1000110111**, συνολικού μήκους 11 bits, άρα περίπου 4 φορές μικρότερη. Βέβαια, στην πράξη, στο τέλος κάθε ακολουθίας προστίθεται και το σχήμα της κωδικοποίησης, ώστε να μπορεί να γίνει η αποσυμπίεση.

Η ακολουθία κωδικοποίησης που παράγεται με τον προηγούμενο αλγόριθμο είναι μοναδική, ώστε να μπορεί να γίνεται η αποσυμπίεση. Έτσι, αν διαθέτουμε το σχήμα της κωδικοποίησης (πίνακας στο πλάι) και την κωδικοποιημένη ακολουθία, τότε μπορούμε να κάνουμε εύκολα την αποσυμπίεση της κωδικοποιημένης ακολουθίας **1000110111**. Διαβάζεται από αριστερά προς τα δεξιά το πρώτο 1. Δεν υπάρχει χαρακτήρας που να αντιστοιχεί σε αυτό. Διαβάζεται και το 0. Ο χαρακτήρας Α αντιστοιχεί στο 10, άρα αντικαθίσταται το 10 με αυτόν. Στη συνέχεια διαβάζουμε το 0, και ο μοναδικός χαρακτήρας που αρχίζει με 0 είναι ο Λ. Όμοια και για τα υπόλοιπα γράμματα.



## Συμπίεση LZW

Πολλά αρχεία, ιδιαίτερα αρχεία χαρακτήρων, έχουν συμβολοσειρές που επαναλαμβάνονται συχνά, για παράδειγμα το άρθρο «τον». Για την

αντιστοιχίσουμε έναν αριθμό σε ολόκληρη τη λέξη (π.χ. το 256 που έχει 2 bytes μήκος), τότε με 2 bytes αντικαθιστούμε παντού τα 5 bytes του άρθρου «τον».

Αυτή είναι η προσέγγιση που ακολουθείται στον αλγόριθμο LZW, ο οποίος παρουσιάστηκε από τους J. Ziv και A. Lempel το 1977, και τροποποιήθηκε από τον T. Welch το 1984.

Ο αλγόριθμος ξεκινά φτιάχνοντας ένα λεξικό όλων των ακολουθιών χαρακτήρων (συμβολοσειρών) που εμφανίζονται στο υπό συμπίεση αρχείο. Στο λεξικό αυτό, που ονομάζεται και *πίνακας αναφορών*, αποθηκεύεται για κάθε συμβολοσειρά ο αριθμός των εμφανίσεων της και δίπλα ένας μοναδικός κωδικός που αντιστοιχεί σε αυτή. Στη φάση της συμπίεσης αντικαθιστούμε κάθε συμβολοσειρά με τον κωδικό της, ο οποίος έχει πολύ μικρότερο μέγεθος από αυτή. Επιπλέον, στο τέλος προσθέτουμε και το λεξικό, ώστε να μπορεί να γίνει η αποσυμπίεση. Για την αποσυμπίεση γίνεται κατ' αρχήν ανάγνωση του λεξικού και αντικατάσταση των κωδικών με τις αντίστοιχες αρχικές συμβολοσειρές. Ο αλγόριθμος LZW χρησιμοποιείται με επιτυχία για αρχεία με μεγάλη κανονικότητα και συχνές επαναλήψεις μεγάλων συμβολοσειρών.

Τα αρχεία τύπου «zip» δημιουργούνται με συμπίεση βασισμένη στον αλγόριθμο LZW.



Οι αλγόριθμοι συμπίεσης χρησιμοποιούνται για την μείωση του μεγέθους που καταλαμβάνουν τα δεδομένα τα οποία είναι αποθηκευμένα στους υπολογιστές. Οι αλγόριθμοι συμπίεσης διακρίνονται σε απωλεστικούς και μη απωλεστικούς. Οι απωλεστικοί αλγόριθμοι συμπιέζουν πολύ περισσότερο, αλλά όμως είναι αδύνατη η αποσυμπίεση της συμπιεσμένης ακολουθίας στην αρχική μορφή. Αντίθετα, με τους μη απωλεστικούς αλγορίθμους, η αντίστροφη διαδικασία της αποσυμπίεσης μας επαναφέρει στην αρχική ακολουθία των δεδομένων. Οι γνωστότεροι μη απωλεστικοί αλγόριθμοι συμπίεσης είναι ο RLE, ο αλγόριθμος Huffman και ο LZW.



Αλγόριθμος Huffman	
Αλγόριθμος LZW	
Αλγόριθμος RLE	
Αλγόριθμος Συμπίεσης	Compression Algorithm
Αποσυμπίεση	Decompression-Extraction
Απωλεστικός Αλγόριθμος	Lossy Algorithm
Μη Απωλεστικός Αλγόριθμος	Lossless Algorithm
Συμπίεση	Compression
Πίνακας Αναφορών	Reference Table

## Ερωτήσεις

- ? Τι εννοούμε με τον όρο «συμπίεση δεδομένων»;
- ? Τι εννοούμε με τον όρο «αποσυμπίεση δεδομένων»;
- ? Ποιες κατηγορίες αλγορίθμων συμπίεσης δεδομένων υπάρχουν. Περιγράψτε τις.
- ? Περιγράψτε τον αλγόριθμο RLE.
- ? Περιγράψτε τον αλγόριθμο Huffman.
- ? Περιγράψτε τον αλγόριθμο LZW.

## Μάθημα 2.4

## Ψηφιακή Παράσταση Ήχου, Εικόνας, Video

Σκοπός του μαθήματος αυτού είναι να εξηγήσει τις μεθόδους και τα σύγχρονα πρότυπα κωδικοποίησης και ψηφιακής αναπαράστασης σύνθετων μορφών ψηφιακών δεδομένων όπως ήχοι, εικόνες και video.

Σκοπός του  
μαθήματος

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

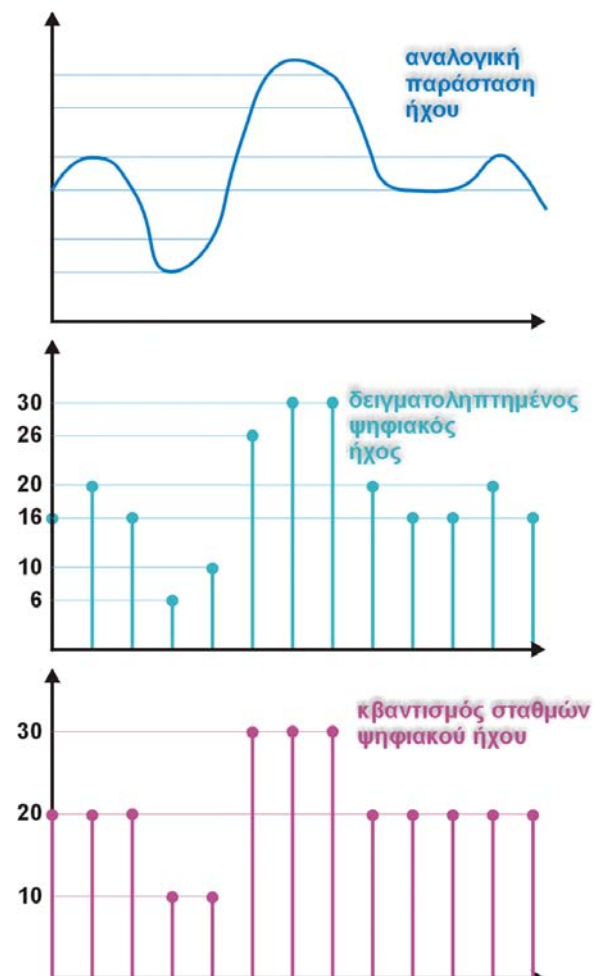
- ♦ Να περιγράφεις τα πρότυπα κωδικοποίησης ήχου
- ♦ Να αναφέρεις τις μορφές αποθήκευσης ψηφιακής εικόνας
- ♦ Να απαριθμείς τι τεχνικές κωδικοποίησης video υπάρχουν

Τι θα μάθεις;

### Ψηφιακή Παράσταση Ήχου

Η ψηφιοποίηση του αναλογικού ήχου γίνεται με την περιοδική λήψη δειγμάτων από το αναλογικό σήμα πολλές φορές το δευτερόλεπτο, η οποία λέγεται *δειγματοληψία* (sampling). Ο αριθμός των δειγμάτων που παίρνουμε ανά δευτερόλεπτο, ώστε ο ψηφιακός ήχος να έχει την ίδια ποιότητα με τον αναλογικό, καθορίζεται από τη μέγιστη συχνότητα που εμφανίζει ο αναλογικός ήχος μας. Ο αριθμός αυτός πρέπει να είναι τουλάχιστο ίσος με το διπλάσιο της μέγιστης συχνότητας του ήχου, σύμφωνα με το θεώρημα του Shannon. Το ανθρώπινο αυτί αντιλαμβάνεται ήχους συχνοτήτων από 20Hz έως 20KHz. Έτσι, για να έχουμε πιστή αναπαραγωγή του αναλογικού ήχου χρειάζονται πάνω από 40.000 δείγματα ανά δευτερόλεπτο.

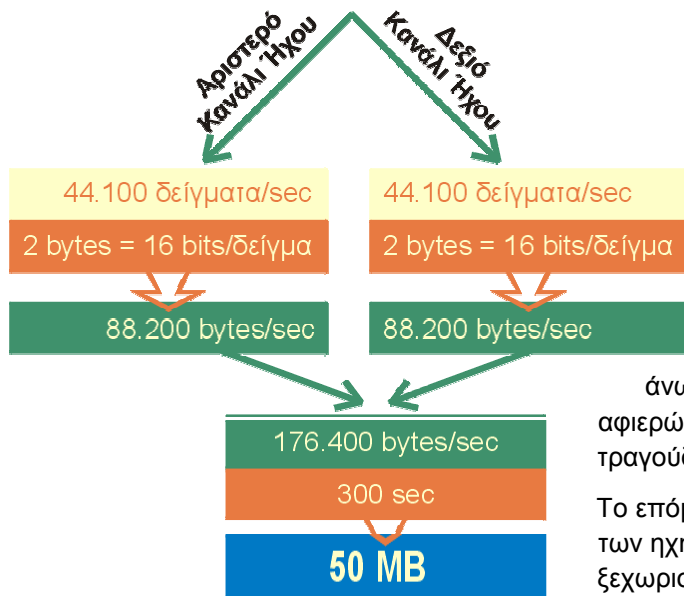
Το πλήθος των δειγμάτων ή αλλιώς ο *ρυθμός* (rate) ή *συχνότητα δειγματοληψίας* (sampling frequency), δεν είναι το μόνο στοιχείο που καθορίζει την κωδικοποίηση του ψηφιακού ήχου. Το κάθε δείγμα αντικατοπτρίζει την ένταση του ήχου για τη στιγμή της δειγματοληψίας στην οποία αντιστοιχεί. Στον αναλογικό ήχο, οποιαδήποτε τιμή έντασης είναι επιτρεπτή. Στον ψηφιακό ήχο όμως, το *πλήθος των bits* (number of bits), που χρησιμοποιούμε για την αποθήκευση του κάθε δείγματος, καθορίζει και τον αριθμό των διαφορετικών τιμών έντασεως που μπορεί να εμφανιστεί. Συνεπώς, κοντινές, αλλά διαφορετικές τιμές αναλογικής έντασης αντιστοιχούν στην ίδια ψηφιακή τιμή. Το φαινόμενο αυτό λέγεται



κβαντισμός (quantization) των σταθμών έντασης του ήχου. Είναι φανερό ότι όσο πιο πολλά bits χρησιμοποιούμε για την αποθήκευση του κάθε δείγματος, τόσο πιο πιστή αναπαράσταση του αναλογικού ήχου πετυχαίνουμε. Αυτός ο τρόπος κωδικοποίησης λέγεται **παλμοκωδική κωδικοποίηση (Pulse Code Modulation)**.

### Στερεοφωνικός ήχος ποιότητας CD

#### 2 κανάλια ήχου



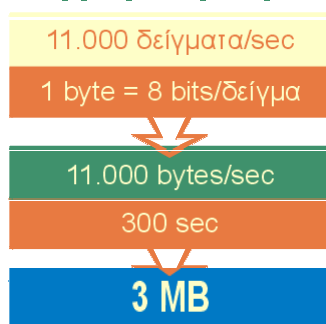
Ήχος σε ποιότητα CD χρειάζεται δείγματα των 16 bits (2 bytes) και ρυθμό δειγματοληψίας 44,1 KHz (χιλιάδες δείγματα ανά δευτερόλεπτο) για κάθε κανάλι. Έτσι για ψηφιακό στερεοφωνικό ήχο (2 κανάλια ήχου) ποιότητας CD έχουμε 176.400 bytes/δευτερόλεπτο· ένα τραγούδι διάρκειας πέντε λεπτών καταλαμβάνει περίπου 50MB.

Αν οι ανάγκες μας σε ποιότητα ήχου είναι μεγάλες, τότε η παλμοκωδική κωδικοποίηση έχει πολύ μεγάλες αποθηκευτικές απαιτήσεις. Σε συνήθεις εφαρμογές πολυμέσων δεν έχουμε απαιτήσεις στερεοφωνικού ήχου ή ακουστικών συχνοτήτων

άνω των 4-5KHz (συχνότητα δειγματοληψίας 11KHz) και αφιερώνουμε 1 byte για κάθε δείγμα. Έτσι για το παραπάνω τραγούδι των 5 λεπτών χρειαζόμαστε το  $\frac{1}{16}$  των 50MB.

Το επόμενο βήμα για να μειώσουμε περισσότερο το μέγεθος των ηχητικών δεδομένων είναι να μην κωδικοποιούμε ξεχωριστά το κάθε δείγμα δίνοντας του π.χ. 8 bits. Είναι πιο οικονομικό να βρίσκουμε τη διαφορά του κάθε δείγματος με το

### Συμπίεση με μειωμένη ποιότητα



προηγούμενο και να κωδικοποιούμε αυτήν. Επειδή ο αριθμός των δειγμάτων ανά δευτερόλεπτο είναι μεγάλος, η πιθανότητα δυο διαδοχικά δείγματα να έχουν κοντινές τιμές έντασης είναι μεγάλη, άρα έχουμε να κωδικοποιήσουμε μικρές διαφορές και δεσμεύουμε λίγα bits, συνήθως 2-3 αντί των 8. Η κωδικοποίηση αυτή λέγεται **διαφορική παλμοκωδική (Differential Pulse Code Modulation)**. Μια παραλλαγή της μεθόδου αυτής που επιτρέπει την ύπαρξη μεγαλύτερων διαφορών λέγεται **προσαρμοστική-διαφορική παλμοκωδική (Adaptive Differential Pulse Code Modulation)** και χρησιμοποιείται

στα αρχεία .wav των Windows.

Όλες αυτές οι τεχνικές αντιμετωπίζουν το ηχητικό σήμα σαν μαθηματική κυματομορφή, αγνοώντας τις ιδιαιτερότητες του ανθρώπινου αυτιού. Έτσι, όλες οι συχνότητες αντιμετωπίζονται ισοδύναμα. Στην πράξη όμως το ανθρώπινο αυτί είναι περισσότερο ευαίσθητο σε κάποια ζώνη συχνοτήτων από κάποια άλλη. Νέες τεχνικές κωδικοποίησης που βασίζονται σε αυτή την απλή παρατήρηση, επιτυγχάνουν πολύ μεγαλύτερα ποσοστά συμπίεσης, χωρίς να υπάρχει ιδιαίτερη απώλεια σε ποιότητα ήχου. Το πρότυπο MPEG-3 (Motion Picture Expert Group) πετυχαίνει συμπίεση έως και 12 φορές ως προς την αρχική μορφή χωρίς εμφανείς απώλειες στην ποιότητα. Τα αρχεία .mp3 είναι κωδικοποιημένα με το πρότυπο αυτό. Αν όμως δεχθούμε συμβιβασμούς στην ακουστική ποιότητα, π.χ. για μονοφωνικό ήχο ποιότητας τηλεφωνικής μετάδοσης, τότε επιτυγχάνεται συμπίεση μέχρι και 100 φορές.

## Ψηφιακή Παράσταση Εικόνας

Μία ψηφιακή εικόνα αποτελείται από μικρές κουκίδες που ονομάζονται *εικονοστοιχεία* (pixels). Στην πιο απλή περίπτωση κάθε εικονοστοιχείο μπορεί να είναι είτε άσπρο είτε μαύρο.

Στο διπλανό σχήμα βλέπουμε μία ασπρόμαυρη ψηφιακή εικόνα. Η εικόνα του σχήματος αποτελείται από 256 γραμμές και 256 στήλες, περιέχει δηλαδή 65.536 διαφορετικά εικονοστοιχεία. Επειδή η φωτογραφία είναι ασπρόμαυρη, κάθε εικονοστοιχείο κωδικοποιείται με ένα bit: το 0 συμβολίζει το μαύρο χρώμα και το 1 το άσπρο. Έτσι, για την αποθήκευση αυτής της φωτογραφίας στον υπολογιστή χρειαζόμαστε 65.536 bits ή ισοδύναμα 8.192 bytes.



Για έγχρωμες εικόνες, κάθε εικονοστοιχείο απαιτεί περισσότερα από 1 bit για την αναπαράσταση του αντίστοιχου χρώματος. Για παράδειγμα, αν έχουμε 16 διαφορετικά χρώματα και κάθε χρώμα εμφανίζει 16 αποχρώσεις, τότε χρειαζόμαστε  $16 \times 16 = 256$  διαφορετικές τιμές. Όλες αυτές μπορούμε να τις αναπαραστήσουμε με ένα byte (8 bits). Αν, για παράδειγμα, έχουμε μια εικόνα μεγέθους  $640 \times 480$ , δηλαδή 307.200 εικονοστοιχεία και για καθένα χρησιμοποιούμε 3 bytes=24bit για την αναπαράσταση του χρώματος, τότε χρειαζόμαστε 921.600 bytes. Τα αρχεία που περιέχουν την εικόνα ασυμπίεστη, ονομάζονται *χαρτογραφικά* (bitmap, BMP).

Το μέγεθος των ασυμπίεστων εικόνων αυξάνει δραματικά με την αύξηση των διαστάσεων της ή του βάθους χρώματος (δηλ. τον αριθμό των bytes για την αναπαράσταση των διαφορετικών χρωμάτων). Η χρησιμοποίηση της μεθόδου συμπίεσης RLE σε εικόνες που είναι σκίτσα ή σχεδιαγράμματα, κυρίως με αποχρώσεις του γκρι, οδηγεί σε σημαντική μείωση του όγκου τους. Η RLE μέθοδος είναι πολύ αποτελεσματική για εικόνες με συνεχόμενες περιοχές εικονοστοιχείων που έχουν το ίδιο χρώμα, καθώς τα αντικαθιστά με ένα κωδικό και το πλήθος τους (βλέπε κεφάλαιο 2.3). Τα αρχεία αυτά είναι τα PCX αλλά και πολλές μορφές TIFF.

Η συμπίεση LZW εφαρμόζεται σε περισσότερο σύνθετες εικόνες, όπως π.χ. έγχρωμες φωτογραφίες. Εκεί, υπάρχει μεγάλη ποικιλία χρωμάτων, που όμως επαναλαμβάνονται πολλές φορές τα ίδια, άρα, όπως είδαμε στο κεφάλαιο 2.3, η κωδικοποίηση τους με λιγότερα bit φέρνει μεγάλα ποσοστά συμπίεσης. Τέτοια συμπίεση υπάρχει στα αρχεία GIF.

Τέλος, αντίθετα με τις προηγούμενες μεθόδους αναπαράστασης που χρησιμοποιούσαν μη απωλεστικούς αλγόριθμους συμπίεσης, υπάρχει μια ευρύτατα διαδεδομένη μέθοδος για εικόνες, που επιτυγχάνει συμπίεση μέχρι και 100-200 φορές σε ορισμένες περιπτώσεις, με μικρή απώλεια της ποιότητας. Η κωδικοποίηση αυτή λέγεται κωδικοποίηση JPEG (Joint Photographic Expert Group). Βασίζεται στην εφαρμογή μιας σειράς πολύπλοκων μαθηματικών μετασχηματισμών στην εικόνα. Για τη συμπίεση ή την αποσυμπίεση από JPEG χρειάζεται αρκετή υπολογιστική ισχύς, κάτι που παλαιότερα ήταν ιδιαίτερα χρονοβόρα. Σήμερα, που υπάρχουν ισχυροί επεξεργαστές, η κωδικοποίηση JPEG δεν είναι πια πρόβλημα, και έχει το πλεονέκτημα της δυνατότητας επιλογής του ποσοστού συμπίεσης που θέλουμε να επιτύχουμε, με αντίστοιχη επιβάρυνση στην ποιότητα της εικόνας. Συνήθως συμπίεσεις 10 έως και 20 φορές, οδηγούν σε εικόνες με μη ορατή διαφορά από την αρχική.



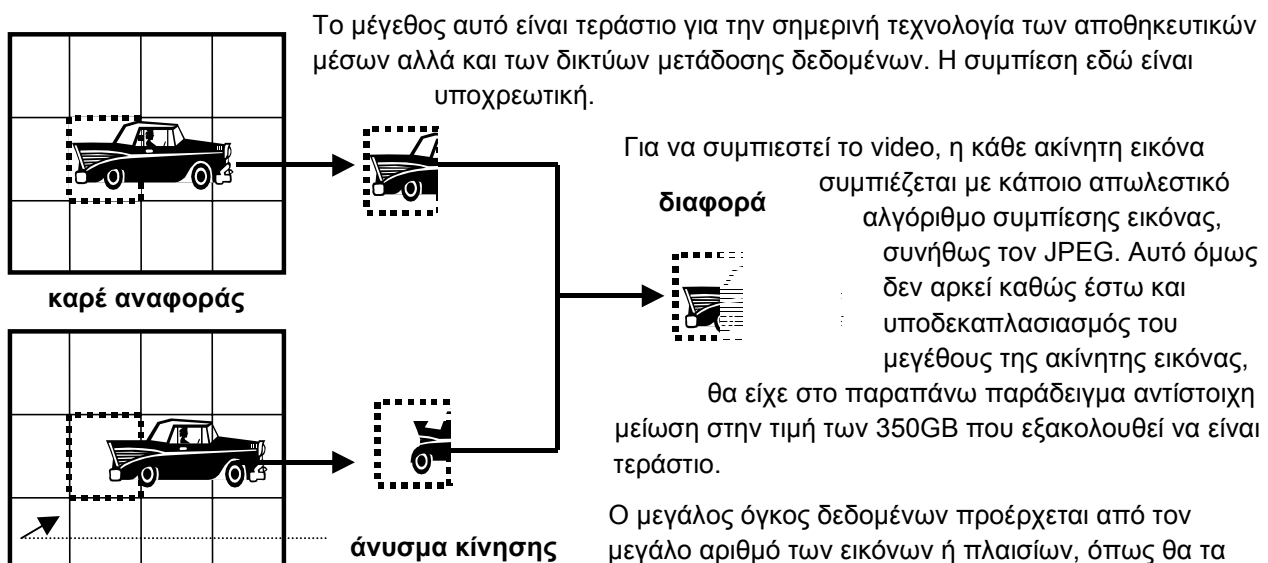
Αντίθετα με την παραπάνω μορφή ψηφιακής αναπαράστασης εικόνας όπου υπάρχει κωδικοποίηση της κάθε εικονοστοιχείου με συγκεκριμένο αριθμό bits, η διανυσματική αναπαράσταση βασίζεται στα γεωμετρικά χαρακτηριστικά των σχημάτων που απεικονίζουν την εικόνα. Έτσι αν π.χ. είχαμε να αναπαρταστήσουμε ένα τρίγωνο, τότε η διανυσματική αναπαράσταση περιλαμβάνει την αποθήκευση των καρτεσιανών συντεταγμένων των τριών κορυφών. Κάθε κορυφή θεωρείται εικονοστοιχείο και αποθηκεύεται όπως πριν. Όμως τώρα έχουμε τρία σημεία που ορίζουν όλο το σχήμα. Η διανυσματική μορφή παράγεται κυρίως από προγράμματα σχεδίασης (CAD – Computer Aided Design). Ο υπολογιστής σχεδιάζει τα γεωμετρικά σχήματα με βάση τις συντεταγμένες που διαβάζει από τέτοια αρχεία

## Ψηφιακή Παράσταση Video

Τα τελευταία χρόνια η εξέλιξη στις μεθόδους συμπίεσης έχει επιτρέψει την αναπαραγωγή κινούμενης εικόνας ή video μέσω των υπολογιστών χαμηλού κόστους. Η κινούμενη εικόνα είναι πιο σύνθετη από την απλή εικόνα ή τον ήχο, καθώς περιέχει εικόνα και ήχο που συγχρονίζονται. Η αρχή πάνω στην οποία στηρίζεται η αναπαραγωγή video είναι ίδια με αυτή του κινηματογράφου. Σε κάθε δευτερόλεπτο εναλλάσσονται αρκετές εικόνες με ταχύτητα και μικροδιαφορές μεταξύ τους, δίνοντας στον άνθρωπο την ψευδαίσθηση της κίνησης. Το ανθρώπινο μάτι δεν μπορεί να αντιληφθεί εύκολα αλλαγές με ρυθμό πάνω από 20 εικόνες/δευτερόλεπτο και επομένως νομίζει πως βλέπει μια συνεχόμενη σκηνή. Στον κινηματογράφο έχουμε 24 εικόνες (ή καρέ όπως λέγονται) ανά δευτερόλεπτο, ενώ στην τηλεόραση κυμαίνονται από 25 καρέ/δευτερόλεπτο (σύστημα PAL στην Ευρώπη) έως και 30 καρέ/δευτερόλεπτο (NTSC στην Αμερική).

Οι συχνότητες αυτές αναφέρονται στο **ρυθμό εναλλαγής των ακίνητων εικόνων** για να σχηματιστεί το video και δεν έχουν σχέση με τη συχνότητα ανανέωσης του πλαισίου που θα δούμε στην τεχνολογία των οθονών.

Αν μια εικόνα διαστάσεων 640×480 εικονοστοιχείων με 24 bits χρώματος για κάθε εικονοστοιχείο, έχει κανονικό μέγεθος 920 KB, τότε για ένα δευτερόλεπτο video με ρυθμό 25 εικόνων ανά δευτερόλεπτο θα χρειαζόμασταν 24MB περίπου. Για μια ταινία 90 λεπτών θα χρειαζόμασταν  $24\text{MB/sec} \times 60\text{sec/min} \times 90\text{min}$  περίπου 3500 GB!!





λέμε στο εξής, ανά δευτερόλεπτο. Η αίσθηση της κίνησης βασίζεται στην ταχύτατη εναλλαγή εικόνων που διαφέρουν πολύ λίγο μεταξύ τους, ώστε το μάτι να μην βλέπει ξαφνικά μεγάλες διαφορές. Η τεχνολογία συμπίεσης σε κινούμενη εικόνα βασίζεται στην απομάκρυνση της πλεονάζουσας πληροφορίας που επαναλαμβάνεται στα διαδοχικά πλαίσια. Η τεχνική που χρησιμοποιείται για την ανίχνευση της πλεονάζουσας πληροφορίας, βασίζεται στην σύγκριση διαδοχικών πλαισίων και στην εύρεση των διαφορών μεταξύ τους. Στη συνέχεια αντί να κωδικοποιηθεί το πλαίσιο, κωδικοποιούμε μόνο τις διαφορές. Το τρέχον πλαίσιο ανακατασκευάζεται από το προηγούμενο και τη διαφορά του με αυτό.

Στην πράξη τα σχήματα κωδικοποίησης είναι αρκετά πιο σύνθετα από την παραπάνω περιγραφή. Η διαδικασία εύρεσης των διαφορών λέγεται *αλγόριθμος εκτίμησης κίνησης* (motion estimation algorithm). Έχουμε δύο διαδοχικά πλαίσια που ονομάζουμε προηγούμενο και τρέχον. Τα δύο πλαίσια διαιρούνται σε στοιχειώδεις μικρές τετραγωνικές περιοχές (blocks). Η κάθε περιοχή του τρέχοντος πλαισίου αναζητείται στο προηγούμενο. Όταν βρεθεί, αντί να κωδικοποιήσουμε την περιοχή με κάποιο αλγόριθμο κωδικοποίησης εικόνας, κωδικοποιούμε το διάνυσμα που δείχνει τις συντεταγμένες του σημείου μετατόπισης, άρα σημαντικά λιγότερη πληροφορία και επομένως επιτυγχάνουμε μεγάλη συμπίεση.

Το πιο διαδεδομένο πρότυπο συμπίεσης για κινούμενη εικόνα είναι το MPEG-2. Για να κωδικοποιήσουμε ένα video σύμφωνα με το πρότυπο αυτό, διαιρούμε το συνολικό αριθμό των 25 ή 30 ή περισσότερων πλαισίων το δευτερόλεπτο σε κατηγορίες.



Ανακεφαλαίωση

Η ψηφιακή αναπαράσταση σύνθετων μορφών δεδομένων που χρησιμοποιούνται κυρίως σε εφαρμογές πολυμέσων γίνεται με την κωδικοποίησή τους σε μορφές που προσφέρουν αρκετή συμπίεση, ώστε να ελαττωθεί το μεγάλο τους μέγεθος. Ο αναλογικός ήχος υπόκειται σε δειγματοληψία και κβαντοποιείται σε ψηφιακές τιμές. Ο ψηφιακός πλέον ήχος κωδικοποιείται με μεθόδους όπως η παλμοκωδική ή η διαφορική παλμοκωδική ή η μέθοδος MPEG-Layer III (.mp3).

Η ψηφιακή αναπαράσταση της εικόνας βασίζεται στην ανάλυσή της σε εικονοστοιχεία. Το κάθε εικονοστοιχείο καταλαμβάνει χώρο ανάλογο του πλήθους των χρωμάτων που χρησιμοποιούνται για την εικόνα. Υπάρχουν τεχνικές κωδικοποίησης που οδηγούν σε απωλεστική συμπίεση (π.χ. JPEG), χωρίς όμως ορατές διαφορές στην εμφάνιση της εικόνας.

Το video αποτελείται από μια σειρά εικόνων που εναλλάσσονται γρήγορα στο χρόνο, ώστε να δημιουργούν την ψευδαίσθηση της κίνησης. Η αναπαράσταση της κινούμενης εικόνας βασίζεται στην κωδικοποίηση των διαφορετικών πλαισίων-εικόνων που αποτελούν το video με τεχνικές αναπαράστασης εικόνων και στην απαλοιφή των κοινών σημείων μεταξύ διαδοχικών πλαισίων, που επαναλαμβάνονται, σχεδόν τα ίδια, πολλές φορές το δευτερόλεπτο.

Γλωσσάριο  
όρων

Αλγόριθμος Εκτίμησης Κίνησης	Motion Estimation Algorithm
Δειγματοληψία	Sampling
Διαφορική Παλμοκωδική Κωδικοποίηση	Differential Pulse Code Modulation
Εικόνα ή Πλαίσιο ή Καρέ	Frame
Κβαντισμός	Quantization
	BMP- Bitmap
Κωδικοποιήσεις	GIF - Graphics Interchange Format
	JPEG - Joint Picture Expert Group
	MPEG - Motion Picture Expert Group
Παλμοκωδική Κωδικοποίηση	Pulse Code Modulation
Προσαρμοστική - Διαφορική	Adaptive Differential Pulse Code
Παλμοκωδική Κωδικοποίηση	Modulation
Ρυθμός	Rate
Συχνότητα Δειγματοληψίας	Sampling Frequency
Εικονοστοιχείο	Pixel

## Ερωτήσεις

- ? Τι λέγεται δειγματοληψία;
- ? Από τι εξαρτάται η ποιότητα του ψηφιακού ήχου;
- ? Τι εννοούμε με τον όρο «κβαντισμό»;
- ? Τι λέγεται παλμοκωδική κωδικοποίηση;
- ? Τι λέγεται διαφορική παλμοκωδική κωδικοποίηση;
- ? Πώς παριστάνεται ο ήχος ψηφιακά;
- ? Πώς παριστάνεται η κινούμενη εικόνα;