lect1PG2

Δυαδική αναπαράσταση δεδομένων (αριθμοί και χαρακτήρες)

lect1PG3

r=10 (δεκαδικό)
Radix=βάση συστήματος.
Κάθε σύστημα βάσης r χρησιμοποιεί r ψηφία.
Πχ το δεκαδικό 0-9
Ο αριθμός 10 δεν είναι ψηφίο του δεκαδικού συστήματος ούτε το 11…… είναι συνδυασμοί των βασικών ψηφίων
12 = 1 δεκάδα και 2 μονάδες
152 = 1 εκατοντάδα 5 δεκάδες και 2 μονάδες
2152 = 2 χιλιάδες 1 εκατοντάδα 5 δεκάδες και 2 μονάδες.

Θέση ψηφίου: Κάθε ψηφίο ανάλογα με τη θέση του απεικονίζει μία ποσότητα.
Πχ 4185

Το 4 αντιστοιχεί σε χιλιάδες. Βάση =10.

10^3 10^2 10^1 10^0
4 1 8 5

Οι δυνάμεις της βάσης =10 αυξάνονται όσο πάμε αριστερά, η μηδενική δύναμη είναι η δεξιότερη.
$10^3 = 1000$
$10^2 = 100$
$10^1 = 10$
$10^0 = 1$

$4185= 4 \times 10^3 + 1 \times 10^2 + 8 \times 10^1 + 5 \times 10^0$

5 μονάδες
8 δεκάδες
1 εκατοντάδα
4 χιλιάδες

ΤΜΗΜΑ ΠΕΜΠΤΗΣ

r -> radix . Η βάση δείχνει πόσα ψηφία (διακριτά) χρησιμοποιεί το σύστημα

r=10 (δεκαδικό) 0,1,2,3,4,5,6,7,8,9

Οι αριθμοί που υπερβαίνουν το 9 είναι συνδυασμοί αυτών των ψηφίων. Πχ ο αριθμός 10 = 1 δεκάδα και 0 μονάδες (συνδυασμός των βασικών ψηφίων 1 και 0)

Ο αριθμός 512 είναι συνδυασμός των βασικών ψηφίων 5,1, και 2. Η ΘΕΣΗ των ψηφίων είναι εκείνη που καθορίζει την αξία ή τιμή του αριθμού.

5,1,2: 215, 512, 152…. Οι αριθμοί αυτοί είναι διαφορετικοί ανάλογα με τη θέση των ψηφίων. ΘΕΣΗ(;;;;;)

512: 5 εκατοντάδες (εκαντοντάδα 100 = 10^2)
1 δεκάδα (δεκάδα 10=10^1)
2 μονάδες (μονάδα 1=10^0)

3512: 3 χιλιάδες (χιλιάδα 1000, 10^3)
5 εκατοντάδες
1 δεκάδα
2 μονάδες

Οι αξίες αρχίζουν από δεξιά: Οι μικρότερες αξίες είναι δεξιά, οι αξίες αυξάνονται προς τα αριστερά.

10^3 10^2 10^1 10^0
3 5 1 2

$3512 = 3x\ 10^3 + 5x10^2 + 1x10^1 + 2\ x10^0$

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Σύστημα αρίθμησης χαρακτηρίζεται από την βάση r (radix).
Βάση είναι το πλήθος των βασικών ψηφίων του συστήματος. Δεκαδικό, r=10 και περιέχει 10 βασικά ψηφιά.
0,1,2,3,4,5,6,7 ,8,9

Ο αριθμός 12 δεν είναι βασικός αριθμός του δεκαδικού συστήματος αλλά είναι συνδυασμός των ψηφίων 1 και 2
Ομοίως, το 21, είναι συνδυασμός των 1, 2
ΕΊΝΑΙ ίδια το 12 και το 21; Χρησιμοποιούν τον ίδιο συνδυασμό. ΔΕΝ είναι ίδια γιατί αυτό που διαφέρει είναι η ΘΕΣΗ των βασικών ψηφίων 1 και 2

12 = 2 μονάδες και μία δεκάδα
21 = 1 μονάδα και δύο δεκάδες
251 = 1 μονάδα, 5 δεκάδες και 2 εκατοντάδες
3251 = 1 μονάδα, 5 δεκάδες, 2 εκατοντάδες και 3 χιλιάδες.

ΘΕΣΗ κάθε βασικού ψηφίου αντιστοιχεί σε μία ποσότητα ή σε ένα βάρος.
ΠΟΙΕΣ είναι αυτές οι ποσότητες ή βάρη στο δεκαδικό σύστημα;
Μονάδες, οι δεκάδες, οι εκατοντάδες, οι χιλιάδες κλπ

Μονάδα: 10^0=1 μηδενική δύναμη του 10
Δεκάδα: 10^1=10 πρώτη δύναμη του 10
Εκατοντάδα: 10^2=100 δεύτερη δύναμη του 10
Χιλιάδα: 10^3=1000 Τρίτη δύναμη του 10

Τα βάρη του δεκαδικού συστήματος είναι δυνάμεις του 10. Το μικρότερο βάρος είναι η μονάδα και τοποθετείται στο δεξί μέρος του αριθμού.

Πχ ο αριθμός 4121

10^3 10^2 10^1 10^0
4 1 2 1

$4121= 4x10^3+1x10^2+2x10^1+1x10^0 = 4000+100+20+1$

lect1PG4

Δυαδικό σύστημα έχει ως βάση το r=2
0 και 1.
Κωδικοποίηση δεδομένων σε 0 και 1.
Π.χ. οι αριθμοί που είναι η πιο αντιληπτή μορφή θα μας απασχολήσουν αρχικά.

11001

Η θέση των ψηφίων αντιστοιχεί σε ποσότητες που είναι δυνάμεις του 2. Η μηδενική δύναμη βρίσκεται τέρμα δεξιά και οι επόμενες προς τα αριστερά αυξάνονται.

ΠΑΡΑΤΗΡΗΣΗ: Αρχικά θα μελετήσουμε **ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ ΑΡΙΘΜΟΥΣ** (χωρίς πρόσημο, άρα θετικοί)

2^4 2^3 2^2 2^1 2^0
1 1 0 0 1

Ο Δεξιότερος άσσος αντιστοιχίζεται σε μονάδα (επειδή $2^0=1$)
Το πρώτο 0 από δεξιά αντιστοιχίζεται σε δυάδα (επειδή $2^1=2$)
Το επόμενο 0 αντιστοιχίζεται σε τετράδα (επειδή $2^2=4$)
Ο επόμενος άσσος αντιστοιχίζεται σε οκτάδα (επειδή $2^3=8$)
Ο αριστερότερος άσσος αντιστοιχίζεται σε δεκαεξάδα (επειδή $2^4=16$)

Ο αριθμός 11001 αποτελείται από 1 μονάδα 0 δυάδες, 0 τετράδες, 1 οκτάδα και 1 δεκαεξάδα=25
$1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4$

_____ **ΤΜΗΜΑ ΠΕΜΠΤΗΣ** _____

Αυτή τη στιγμή αναφερόμαστε σε ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ αριθμούς.
Αργότερα οι αριθμοί θα εκφράζονται σε byte ή πολλαπλάσια αυτού (1 byte = 8 bit, δηλαδή 8 μηδενικά ή άσσους)

Ποιος δεκαδικός αριθμός είναι ο 101011;

2^5 2^4 2^3 2^2 2^1 2^0
1 0 1 0 1 1

$1x2^5+1x2^3+1x2^1+1x2^0=32+8+2+1=43$

ΠΑΡΑΤΗΡΗΣΕΙΣ: Μέχρι να αναφερθεί κάτι διαφορετικό μιλάμε για αριθμούς **ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ (unsigned)**. Δεν έχουν πρόσημο και είναι πάντα θετικοί.
Οι αριθμοί κανονικά γράφονται σε ένα ή πολλαπλά byte (1byte=8 bit, δηλαδή 8 μηδενικά ή μονάδες). Προς το παρόν θα γράφουμε τους αριθμούς σε όσα bit χρειαστεί. Π.χ. ο αριθμός 1000110 είναι 7 bit

$(1000110)_2$
Αναζητούμε τον αντίστοιχο δεκαδικό.

2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	0	0	1	1	0

$1x2^6+1x2^2+1x2^1 = 64+4+2=70$

lect1PG5

Δυαδικό από 0 έως 15

Πίνακας 1.1 Οι αριθμοί από 0 έως 15 στο δεκαδικό, δυαδικό, οκταδικό και δεκαεξαδικό σύστημα αρίθμησης

Δεκαδικό	Δυαδικό	Οκταδικό	Δεκαεξαδικό
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

2^3	2^2	2^1	2^0	
0	1	0	1	= 1 x 2^0 + 0 x 2^1 + 1 x 2^2 + 0 x 2^3= 0+4+0+1=5

lect1PG6

ΔΥΝΑΜΕΙΣ ΤΟΥ ΔΥΟ

$2^0 = 1$	$2^8 = 256$
$2^1 = 2$	$2^9 = 512$
$2^2 = 4$	$2^{10} = 1024=1K$
$2^3 = 8$	$2^{11} = 2048=2K$
$2^4 = 16$	$2^{12} = 4096=4K$
$2^5 = 32$	$2^{13} = 8192=8K$
$2^6 = 64$	$2^{14} = 16384=16K$
$2^7 = 128$	$2^{15} = 32768=32K$

Επίσης, $1M=2^{20}$, $1G=2^{30}$. $1T=2^{40}$, $1P=2^{50}$

2^{10}=1K (το 1K είναι ΑΡΙ0ΜΟΣ όχι μονάδα μέτρησης) =1024
1KByte = 1024 bytes. Το byte είναι η βασική μονάδα μέτρησης του υπολογιστή (1 byte=8 bit), όπου κάθε bit είναι ένα μηδενικό ή μία μονάδα.

Στην καθημερινότητα, όταν λέμε ΚΙΛΟ εννοούμε χιλιάδα. 1 κιλό = 1000 γραμμάρια.
Η χιλιάδα του δεκαδικού συστήματος είναι το 10^3, δηλαδή μία ακέραια δύναμη του 10 που είναι η βάση
Στο δυαδικό σύστημα, δεν υπάρχει δύναμη του 2 ακέραια, στην οποία αν υψώσουμε το 2 παίρνουμε 1000. 2^x=1000, το x δεν είναι ακέραιο. Όμως η πλησιέστερη δύναμη του 2 κοντά στο 1000 είναι η 10 και 2^{10}=1024. Για αυτό τον λόγο το kilo στους υπολογιστές είναι 1024=2^{10}=1K

_____ ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

1 kilobyte=1024 bytes,
1 Megabyte= 1024 Kbyte = 210x210= 220 byte

1K= 210=1024

3K πως θα το εκφράσουμε σε δύναμη του 2;

3 x 210
Δηλαδή, το Κ εκφράζεται ως ΑΚΕΡΑΙΑ δύναμη του 2 αλλά το 3 ΌΧΙ, δηλαδή ΔΕΝ ΥΠΑΡΧΕΙ ακέραια δύναμη του 2 που να δίνει 3
Αν πάμε να λύσουμε την εξίσωση 2^x=3, το x δεν είναι ακέραιο. Άρα δεν μπορούμε να εκφράσουμε ως ακέραια δύναμη του 2 το 3K

Στην καθημερινότητα, 1 Kilo αντιστοιχεί σε χιλιάδα.
Στο δυαδικό σύστημα δεν υπάρχει ακέραιος x, στον οποίο να υψώσουμε το 2 και να πάρουμε 1000
2^x=1000 δεν λύνεται για ακέραιο x
Επειδή η ΑΚΕΡΑΙΑ δύναμη του 2 που δίνει αριθμό πλησιέστερο στο 1000 είναι η δέκατη, και η τιμή είναι 1024, λέμε ότι το kilo στον υπολογιστή είναι το 1024.

_____ ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

1 Kbyte
1K είναι αριθμός =1024

Στην καθημερινότητα 1 Κιλό = 1000 γραμμάρια ή 10^3 γραμμάρια.
Δηλαδή, μια ΑΚΕΡΑΙΑ δύναμη του 10, η Τρίτη, αναπαριστά αυτό που λέμε κιλό.
Στους υπολογιστές που χρησιμοποιούν δυαδικό σύστημα
ΔΕΝ υπάρχει ΑΚΕΡΑΙΑ δύναμη του 2 η οποία δίνει 1000. Διαφορετικά, η εξίσωση 2^x=1000, δεν έχει ακέραια λύση.
ΌΜΩΣ, η ακέραια δύναμη του 2 που δίνει τιμή κοντά στο 1000 είναι η 10η. **Άρα το kilo του υπολογιστή είναι το 2^{10}=1024**

lect1PG7

ΜΕΤΑΤΡΟΠΗ ΔΕΚΑΔΙΚΟΥ ΣΕ ΔΥΑΔΙΚΟ

Μέθοδος διαιρέσεων: Ξεκινάμε από έναν δεκαδικό αριθμό και διαιρούμε διαδοχικά με το 2 λαμβάνοντας ένα ακέραιο πηλίκο και ένα υπόλοιπο. Το πηλίκο διαιρείται ξανά με το 2 και παράγει ένα νέο πηλίκο και ένα νέο υπόλοιπο. Η διαδικασία σταματά όταν το πηλίκο μας γίνει 0. Τότε, διαβάζουμε τα υπόλοιπα ανάποδα από τη σειρά παραγωγής τους.

Δεκαδικός Αριθμός	Ακέραιο Πηλίκο διαίρεσης με το 2	Υπόλοιπο διαίρεσης με το 2
50	25	0
25	12	1
12	6	0
6	3	0
3	1	1
1	0	1

Διαβάζω τα υπόλοιπα με σειρά αντίστροφη (από κάτω προς τα επάνω) 110010.
Επαλήθευση ότι 110010=50
110010

2^5 2^4 2^3 2^2 2^1 2^0
1 1 0 0 1 0

= 1x 25 +1 x 24 +1x21 = 32+16+2=50

Για να γράψουμε τον αριθμό 50 (*μη προσημασμένη μορφή*) απαιτούνται 6 bit. Στην πράξη όμως, όλοι οι αριθμοί γράφονται σε μονάδες πολλαπλάσιες του byte. ΔΕΝ υπάρχει αποθήκευση 6 bit θα ξοδευτεί 1 byte για αυτό τον αριθμό. Μετά τα εισαγωγικά μαθήματα, όλες οι αναπαραστάσεις θα είναι σε επίπεδο byte.

Signed integer, *unsigned integer*: Ο signed ξοδεύει ένα Bit (το αριστερότερο) για πρόσημο, **1 για αρνητικό πρόσημο**, 0 **για θετικό**. Ο unsigned δεν έχει πρόσημο με συνέπεια το αριστερότερο bit να αντιστοιχεί σε *ΠΟΣΟΤΗΤΑ*.

2η μέθοδος: **Βρίσκουμε την μεγαλύτερη δύναμη του 2 που «χωράει» στον ζητούμενο αριθμό. Επαναλαμβάνουμε μία σειρά αφαιρέσεων δίνοντας τιμές στους άλλους συντελεστές**.

Πχ. 50. Η μεγαλύτερη ακέραια δύναμη είναι η 5η επομένως πρέπει να βρω τους συντελεστές (0 και 1) που αντιστοιχίζονται στις δυνάμεις από την 5η ως την μηδενική, δηλαδή 6 συντελεστές (6 bit)

1 1 0 0 1 0

_ _ _ _ _ _

Το 32 χωράει στο 50 και περισσεύουν 50-32=18.
Ο επόμενος συντελεστής αντιστοιχεί σε 16άδες. Το ερώτημα είναι: Υπάρχει 16άδα στο 18; ΝΑΙ και περισσεύουν 18-16=2. Αφού η απάντηση είναι ΝΑΙ ο συντελεστής είναι 1.
Ο επόμενος συντελεστής αντιστοιχεί σε 8άδες. Υπάρχει 8άδα στο 2; ΌΧΙ άρα βάζουμε 0
Ο επόμενος συντελεστής αντιστοιχεί σε 4άδες. Υπάρχει 4άδα στο 2; ΌΧΙ άρα βάζουμε 0
Ο επόμενος συντελεστής αντιστοιχεί σε 2άδες. Υπάρχει 2άδα στο 2; ΝΑΙ άρα βάζουμε 1 Υπόλοιπο 2-2=0

TIP: Οι άρτιοι αριθμοί έχουν δεξιότερο bit =0, οι περιττοί =1.

Ξεκινάμε από την μεγαλύτερη δύναμη του 2 στην οποία χωράει ο αριθμός μας. Αν αυτή είναι η 2^x τότε χρειαζόμαστε **x+1 bit**

Π.χ. για το 50 η μεγαλύτερη δύναμη είναι το 32 = 25, άρα x=5 και θέλουμε 5+1 = 6 bit
Π.χ. για το 70 η μεγαλύτερη δύναμη είναι το 64 = 26, άρα x=6 και θέλουμε 6+1 = 7 bit
Π.χ. για το 130 η μεγαλύτερη δύναμη είναι το 128 = 27, άρα x=7 και θέλουμε 7+1 = 8 bit

_____ **ΤΜΗΜΑ ΠΕΜΠΤΗΣ**_____

Μέθοδος διαιρέσεων: Ξεκινάμε από έναν δεκαδικό αριθμό και διαιρούμε διαδοχικά με το 2 λαμβάνοντας ένα ακέραιο πηλίκο και ένα υπόλοιπο. Το πηλίκο διαιρείται ξανά με το 2 και παράγει ένα νέο πηλίκο και ένα νέο υπόλοιπο. Η διαδικασία σταματά όταν το πηλίκο μας γίνει 0. Τότε, διαβάζουμε τα υπόλοιπα ανάποδα από τη σειρά παραγωγής τους.

Έστω ότι θέλουμε να βρούμε το 75

Αριθμός	Ακέραιο πηλίκο διαίρεσης με το 2	Υπόλοιπο
75	37	1
37	18	1
18	9	0
9	4	1
4	2	0
2	1	0
1	0	1

1001011 =75 = 1x26+1x23+1x21+1x20

ΣΥΜΠΕΡΑΣΜΑ: Ο αριθμός 75 «χώρεσε» σε 7 bit. Σε επίπεδο byte θα γράφαμε 01001011 (8 bit)

ΔΕΥΤΕΡΗ ΜΕΘΟΔΟΣ
Βασίζεται στον ε**ντοπισμό της μεγαλύτερης δύναμης του 2 η οποία χωράει στον ζητούμενο αριθμό. Στη συνέχεια, γνωρίζοντας πόσα Bit απαιτεί ο αριθμός συμπληρώνουμε τις υπόλοιπες θέσεις εξετάζοντας**

αν οι δυνάμεις (οι επόμενες προς τα πίσω δυνάμεις) χωράνε στα υπόλοιπα που θα παράγονται.

Αν βρούμε ότι η μεγαλύτερη δύναμη του 2 που χωράει στον ζητούμενο αριθμό είναι η x, τότε απαιτούνται x+1 bit.
Π.χ., στο 75 η δύναμη αυτή είναι η x=6 (2^6=64). Άρα ο αριθμός απαιτεί 6+1=7 bit

2^6 2^5 2^4 2^3 2^2 2^1 2^0

1 0 0 1 0 1 0

Ο συντελεστής της 6ης δύναμης του 2 είναι 1.

5η δύναμη: Αφαιρούμε από το 75 τα 64 = 11 (απομένει να προσθέσουμε άλλα 11 στο 64).
Χωράει 32αδα στο 11 (ΟΧΙ) άρα ο συντελεστής της 5ης δύναμης του 2 είναι 0

4η: Χωράει 16 στο 11; ΟΧΙ άρα βάζουμε 0 στον συντελεστή της τέταρτης δύναμης

3η: Χωράει 8 στο 11; ΝΑΙ και απομένουν άλλα (11-8) να προστεθούν. Άρα βάζουμε 1 στον συντελεστή της τρίτης δύναμης

2η: Χωράει 4 στο 3; ΟΧΙ Βάζουμε 0 στον συντελεστή της 2ης δύναμης

1η: Χωράει 2 στο 3; ΝΑΙ Απομένει να προσθέσουμε 3-2=1

100000011 =259 ΣΕ ένα byte ΔΕΝ χωράει

00000001 00000011

Ο μεγαλύτερος unsigned integer αν η αποθήκευση είναι σε 2 bytes

11111111 11111111 = 216-1=65535

111=7 =23-1

0111111 11111111= 32767

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Μας δίνεται ο αριθμός 84 και θέλουμε να βρούμε τη δυαδική του μορφή.

Μέθοδος διαδοχικών διαιρέσεων: Ξεκινάμε από τον ζητούμενο αριθμό. Διαιρούμε διαδοχικά με το 2. Κάθε φορά παράγεται ένα ακέραιο πηλίκο και ένα υπόλοιπο το οποίο είναι 0 ή 1. Το νέο πηλίκο διαιρείται με το 2 και η διαδικασία επαναλαμβάνεται, μέχρι να φτάσουμε σε πηλίκο 0. Στη συνέχεια διαβάζουμε τα υπόλοιπα ΜΕ ΣΕΙΡΑ ΑΝΤΙΣΤΡΟΦΗ ΑΠΌ ΤΗ ΣΕΙΡΑ ΠΑΡΑΓΩΓΗΣ ΤΟΥΣ

Αριθμός	Ακέραιο Πηλίκο διαίρεσης με το 2	Υπόλοιπο
84	42	0
42	21	0
21	10	1
10	5	0
5	2	1
2	1	0
1	0	1

Διαβάζουμε τα υπόλοιπα με την αντίστροφη σειρά:
$1010100 = 1x2^6 + 1x2^4 + 1x2^2 = 64 + 16 + 4 = 84$

2η μέθοδος: Αρχικά βρίσκουμε την μέγιστη δύναμη του 2 που χωράει στον ζητούμενο αριθμό. Αν η δύναμη αυτή είναι η x, τότε απαιτούνται x+1 bit. Στο παράδειγμα έχουμε x=6. Άρα απαιτούνται 7 bit.
Γράφουμε με κενά τις x+1 θέσεις συμπληρώνοντας την πρώτη θέση από αριστερά με 1 (η x-οστή δύναμη).

1 0 1 0 1 0 0
_ _ _ _ _ _ _

Συμπληρώνουμε τις υπόλοιπες x θέσεις.

Έχουμε βάλει στον αριθμό μας μία 64άδα. Άρα απομένει να βάλουμε άλλα 20 (84-64) στις υπόλοιπες θέσεις. Άρα πρέπει να συμπληρώσουμε τις θέσεις των δυνάμεων από την πέμπτη ως τη μηδενική.
5η δύναμη: ΕΡΩΤΗΜΑ χωράει το 32 = 25 στο 20; ΟΧΙ άρα γράφω 0
4η δύναμη: ΕΡΩΤΗΜΑ χωράει το 16 = 24 στο 20; ΝΑΙ άρα γράφω 1, άρα έχω προσθέσει στον αριθμό μου μία 64άδα και μία 16άδα και μένει να προσθέσω άλλα 20-16=4
3η δύναμη: ΕΡΩΤΗΜΑ χωράει το 8 = 23 στο 4; ΟΧΙ άρα γράφω 0
2η δύναμη: ΕΡΩΤΗΜΑ χωράει το 4 = 22 στο 4; ΝΑΙ άρα γράφω 1. Περισσεύει 0

Ξέρουμε πάντοτε τι τιμή έχει το δεξιότερο ψηφίο ενός δυαδικού.
Ζυγοί- > δεξιότερο bit =0
Μονοί -> 1

Αυτό συμβαίνει γιατί όλες οι δυνάμεις του 2 πλην της μηδενικής είναι ζυγές.

lect1PG8

Αριθμοί λίγο μικρότεροι από ακριβείς δυνάμεις του 2

Εκμεταλλευόμαστε το γεγονός ότι ο αριθμός 2^n-1 γράφεται με n μονάδες και αφαιρούμε κατάλληλα

30 < 32
1022 <1024
32760 <32768
11 = 3 = 2^2-1
111 = 7 = 2^3-1
1111 = 15 = 2^4-1

Ένας αριθμός εκφρασμένος με n μονάδες είναι ίσος με 2^n-1

32 = 25

31 = 2^5-1 = 11111 Αν θέλω να βρω το 30 δεν έχω παρά να γράψω το 31 και να αφαιρέσω 1, δηλαδή 11110

1020 .
Ξέρω ότι 2^{10}=1024 άρα 1111111111 =1023
Για να βρω το 1020 αφαιρώ από το 1111111111 τον αριθμό 3, δηλαδή θέτω 0 τα δύο τελευταία ψηφία. Τελικά 1020= 1111111100

32760: Γράφω 15 άσσους που είναι ο αριθμός 32767 και αφαιρώ 7

32767= 111111111111111
32760 = 111111111111000

10 άσσοι = 1023
1024 = 1 άσσος και 10 μηδενικά 10000000000

32768 = 2^{15} = 1000000000000000 (οι δυνάμεις ξεκινούν από το 0 άρα το 16ο ψηφίο από δεξιά προς τα αριστερά είναι η 15η δύναμη του 2)

32767+1=32768

111111111111111 +
000000000000001 =

1000000000000000

Ένας άσσος ακολουθούμενος από n μηδενικά = 2^n, δηλαδή ακριβής δύναμη του 2.
n άσσοι είναι 2^n-1

Ποιος αριθμός σε δυαδική μορφή είναι το 4090;

4095 = 111111111111 αφαιρούμε 5 και έχουμε 4090 = 111111111010

Ποιος αριθμός σε δυαδική μορφή είναι το 8180;

8191 = 1111111111111
Αφαιρώ 11, δηλαδή 8+2+1 άρα 8180 = 1111111110100

Όταν ψάχνουμε αριθμούς μικρότερους λίγο από ακριβείς δυνάμεις του 2 κάνουμε τα εξής:
- Βρίσκουμε την ακριβή δύναμη του 2 που είναι λίγο μεγαλύτερη από

τον αριθμό που ψάχνουμε πχ 2^n αυτός ο αριθμός
- Γράφουμε το 2^n-1, το οποίο είναι n μονάδες.
- Αφαιρούμε κατάλληλα , ώστε να φτάσουμε στον επιθυμητό αριθμό

_____ ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

30 < 25 = 32
1022 < 210 =1024

Ο αριθμός 15 = 16-1 = 2^4-1. Ο αριθμός 15= 1111, δηλαδή δεδομένου ότι το 16 είναι 24, το 15 είναι 4 μονάδες.
16 = 1 μονάδα ακολουθούμενη από 4 μηδενικά. Δηλαδή το 16 απαιτεί 5 bit για την αναπαράστασή του.

Αριθμός-ακριβής δύναμη του 2, δηλαδή 2^n: γράφεται με 1 μονάδα ακολουθούμενη από n μηδενικά, δηλαδή απαιτεί n+1 bits. Από την άλλη, ο αριθμός ακριβής δύναμη τους 2 μείον 1, δηλαδή 2^n-1 γράφεται με n μονάδες, δηλαδή απαιτεί n bits.

2^3 = 8 = 1000, ενώ 2^3-1=7= 111
2^{10}=1024 = 10000000000, ενώ 2^{10}-1=1023= 1111111111

Παραδείγματα

29 <32, συγκεκριμένα το 31 είναι η πλησιέστερη τιμή προς το 32, η οποία μπορεί να γραφτεί με 5 μονάδες. Συνεπώς μπορούμε να γράψουμε το 31 και να αφαιρέσουμε 2 για να φτάσουμε στο 29.

31=111<u>1</u>1. Άρα αν ο υπογραμμισμένος άσσος γίνει 0, ο αριθμός που προκύπτει είναι το 29 Δηλ. 29 = 11101

Το 25=31-6.

31=11111. Αφαιρώ 6 άρα μηδενίζω τις μονάδες που αντιστοιχούν στα βάρη 2, 4.
Άρα 25=11001

245

255 = 11111111 (επειδή 28=256, άρα 255 είναι γραμμένο με 8 άσσους). Άρα αφαιρούμε 10=2+8
245 =11110101

32764. Βλέπουμε ότι 32768=215, άρα 32767 = 111111111111111. Από το 32767 αφαιρώ 2+1=3
Άρα 32764=111111111111100

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

1η παρατήρηση: Ένας αριθμός που ισούται με 2^n-1 είναι γραμμένος στο δυαδικό σύστημα με n μονάδες
Παραδείγματα
11 = 3 = 2^2-1 (δύο μονάδες δίνουν τον αριθμό 22-1=3)

111 = 7 = 2^3-1 (τρεις μονάδες δίνουν τον αριθμό 23-1=7)
11111111 = 255 = 2^8-1 (οκτώ μονάδες δίνουν τον αριθμό 28-1=255)

Για να γράψουμε τον αριθμό 2^n-1 χρειαζόμαστε n bit, τα οποία είναι όλα ίσα με 1.

2^n Ένας αριθμός που ισούται με 2^n

Είναι γραμμένος με 1 μονάδα ακολουθούμενη από n μηδενικά
Πχ 10 = 2 (μία μονάδα ακολουθούμενη από ένα μηδενικό. 2^1=2)
100 = 4 (μία μονάδα ακολουθούμενη από δύο μηδενικά. 2^2=4)
1000 = 8 (μία μονάδα ακολουθούμενη από τρία μηδενικά. 2^3=8)
100000000 = 256 (μία μονάδα ακολουθούμενη από οκτώ μηδενικά. 2^8=256)

Για να γραφτεί ο αριθμός 2^n απαιτούνται n+1 bit (επειδή οι δυνάμεις ξεκινούν από την μηδενική)

ΣΥΝΟΛΙΚΑ
7= 111 (τρία bit, 2^3-1)
8 = 1000 (τέσσερα bit, 2^3=8)

511 = 111111111 (εννέα bit, 2^9-1)
512 = 1000000000 (δέκα bit, 2^9)

ΑΡΙΘΜΟΙ μικρότεροι από ακέραια δύναμη του 2

Έστω ότι θέλουμε τον αριθμό 27.
Γνωρίζουμε ότι το 31 είναι 5 μονάδες 11111
Από το 31 για να πάμε στο 27 αφαιρούμε 4. Για να αφαιρέσουμε 4 μετατρέπουμε τη μονάδα που αντιστοιχεί στο βάρος 4 = 2^2, σε 0
Τελικά 27=11011
Έστω ότι θέλουμε το 25
31= 11111 (25=31-6, άρα αφαιρώ τετράδες και δυάδες αφού 2+4=6).
Τελικά 11001

Έστω ότι θέλουμε τον αριθμό 1010.

1111111111 =1023

Αφαιρώ 13 = 8+4+1
Τελικά 1010= 1111110010

1M -4: 11111111111111111100

lect1PG9

Αριθμοί λίγο μεγαλύτεροι από ακριβείς δυνάμεις του 2

Εκμεταλλευόμαστε ότι το 2^n είναι μία μονάδα ακολουθούμενη από n μηδενικά και προσθέτουμε κατάλληλα

$18 = 2^4 + 2$
$2050 = 2^{11} + 2$
$32770 = 2^{15} + 2$

Άρα, γράφω το 2^n που είναι μία μονάδα ακολουθούμενη από n μηδενικά και μετά αλλάζω κάποια από τα μηδενικά σε μονάδες, άρα προσθέτω την κατάλληλη ποσότητα.

18: Γράφω το 2^4=16, ως 10000 και προσθέτω 2, δηλαδή αλλάζω από 0 σε 1 το δεύτερο bit από δεξιά. Τελικά, 18=10010

2055: Ξέρουμε ότι 2^{11}=2048, δηλαδή μία μονάδα ακολουθούμενη από 11 μηδενικά: 100000000000
Επειδή το 2055 απέχει κατά 7 από το 2048, αλλάζω τα bit που αντιστοιχούν στις ποσότητες 4,2,1 δηλαδή τα τρία τελευταία.
Τελικά 2055 = 100000000111

4120: Γράφουμε το 4096=1000000000000 και μας χρειάζονται άλλα 24 = (16+8). Άρα αλλάζουμε από 0 σε 1 τα ψηφία που αντιστοιχούν στην τέταρτη και στην Τρίτη δύναμη του 2. Τελικά 4120= 1000000011000

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

8=1000
16 =10000

Έστω ότι μας ζητείται ο αριθμός 19 = 16+3=16+2+1
Άρα γράφω το 16 10000 και προσθέτω 3, δηλαδή αλλάζω τα δύο τελευταία bit από 0 σε 1. Άρα 19=10011

135.
128=27=10000000. Πρέπει να προσθέσω 7=4+2+1. Τελικά 135=10000111.

4102: 4096=212=1000000000000 = 1000000000110

_____ ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Έστω ότι θέλουμε τον αριθμό 38.

32 = 100000. Σε αυτό προσθέτουμε 4+2=6. Τελικά 38= 100110.

263 = 100000111

256= 100000000
Πρέπει να προσθέσω 4+2+1=7

2048= 100000000000. Προσθέτω 13= 8+4+1.
2061= 100000001101

lect1PG10

ΚΛΑΣΜΑΤΙΚΟ ΜΕΡΟΣ ΑΡΙΘΜΟΥ

Οι δυνάμεις του 2 αυξάνονται από δεξιά προς αριστερά ξεκινώντας από το 0. Τι συμβαίνει όταν υπάρχει υποδιαστολή;

			2^1	2^0	2^{-1}	2^{-2}
1	0	1	0	0.	1	1

Δεξιά της υποδιαστολής οι δυνάμεις μειώνονται

Πως θα βρούμε τον αριθμό;
Από την υποδιαστολή και αριστερά κάνουμε αυτά που γνωρίζουμε και μετατρέπουμε τον αριθμό χωριστά. Δηλαδή 10100 = 20 ΑΚΕΡΑΙΟ ΜΕΡΟΣ
Το κλασματικό μέρος μετατρέπεται ξεχωριστά:

$1 \times 2^{-1} + 1 \times 2^{-2} = ½ + ¼ = 0,75$

Επειδή $1x2^{-1} = 1/2^1$ και $1x\ 2^{-2} = 1/2^2$

Τελικά ο αριθμός είναι 20,75

ΜΕΤΑΤΡΟΠΗ από δεκαδικό σε δυαδικό για αριθμούς με κλασματικό μέρος

ΑΛΓΟΡΙΘΜΟΣ
- Το ακέραιο μέρος μετατρέπεται ξεχωριστά και το κλασματικό ξεχωριστά
- Το ακέραιο μέρος μετατρέπεται με έναν από τους τρόπους που γνωρίζουμε
- Το κλασματικό μέρος απαιτεί μία σειρά πολλαπλασιασμών με το 2 έως ότου το νέο γινόμενο που θα πάρουμε να είναι 0 και επειδή αυτό δεν είναι βέβαιο, μέχρι να φτάσουμε στην επιθυμητή ακρίβεια.

Π.χ. 20.625

Το 20 = 10100

Αριθμός	Γινόμενο (x2)	Κλασματικό μέρος	Συντελεστής (ακέραιο μέρος πολ/μού)
0.625	1.25	0.25	1
0.25	0.50	0.50	0
0.50	1	0	1

Άρα 0.625 = 101 (**οι συντελεστές διαβάζονται με τη σειρά από πάνω προς τα κάτω**)
Σε κάθε γινόμενο, το κλασματικό μέρος (Τρίτη στήλη) πάει στον επόμενο πολ/μό
10100.101=20.625

2ο παράδειγμα: 20.4
Το 20 = 10100

Αριθμός	Γινόμενο	ΚΜ	Συντελεστής
0.4	0.8	0.8	0
0.8	1.6	0.6	1
0.6	1.2	0.2	1
0.2	0.4	0.4	0

Άρα, με **ακρίβεια τεσσάρων bit** στο κλασματικό μέρος 20.4= 10100.0110
Με ακρίβεια 6 bit στο κλασματικό μέρος 20.4= 10100.011001
Με ακρίβεια 8 bit στο κλασματικό μέρος 20.4= 10100.01100110

Πρόσθεση – Αφαίρεση
(Μη προσημασμένους)
Αποθήκευση ακέραιων χωρίς πρόσημο και με πρόσημο.
Κινητή υποδιαστολή (REAL)
Hardware

_____ ΤΜΗΜΑ ΠΕΜΠΤΗΣ _____

11011.101 (ποιος είναι ο δεκαδικός αριθμός)

2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	1	0	1	1.	1	0	1

Το ακέραιο μέρος είναι το 27. Το κλασματικό είναι $1x2^{-1} + 1x2^{-3}$ $2^{-1} = 1/2^1 = 1/2 = 0.5$ και $2^{-3}=1/2^3=1/8=0.125$. Άρα τελικά το κλασματικό μέρος είναι 0.625 και ο αριθμός είναι το 27.625.

ΜΕΤΑΤΡΟΠΗ ΑΠΌ ΔΕΚΑΔΙΚΟ ΣΕ ΔΥΑΔΙΚΟ ΤΟΥ ΚΛΑΣΜΑΤΙΚΟΥ ΜΕΡΟΥΣ:
Βασίζεται σε μία διαδικασία πολλαπλασιασμών, η οποία δεν είναι βέβαιο ότι θα τερματίσει. Αν δεν τερματίζει, τότε σταματάμε σε μία *επιθυμητή ακρίβεια.*

Παράδειγμα 16.4: Για να βρούμε τη δυαδική τιμή μετατρέπουμε χωριστά το ακέραιο και το κλασματικό μέρος. 16= 10000. Για το 0.4

Αριθμός	Ακέραιο γινόμενο με το 2	Κλασματικό γινόμενο με το 2	Συντελεστής
0.4	0	0.8	0
0.8	1	0.6	1
0.6	1	0.2	1
0.2	0	0.4	0

Επαναλαμβάνεται ένας κύκλος. Οι πολλαπλασιασμοί θα σταματούσαν αν το κλασματικό γινόμενο ήταν 0
Άρα με *επιθυμητή ακρίβεια 4 bit*, ο αριθμός 0.4= 0.0110 (διαβάζουμε από πάνω προς τα κάτω)
Με *επιθυμητή ακρίβεια 7 bit* θα γράφαμε 0.4 = 0.0110011
ΤΕΛΙΚΑ 16.4=10000.0110 με ακρίβεια 4 bit

2ο παράδειγμα: 16.125

Αριθμός	Ακέραιο γινόμενο με το 2	ΚΓ με το 2	Συντελεστής
0.125	0	0.25	0
0.25	0	0.5	0
0.5	1	0	1

0.125=0.001
Φυσικά το 16.125= 10000.001

_____ ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Για να μετατρέψουμε στο δυαδικό σύστημα έναν αριθμό κλασματικό ακολουθούμε μία σειρά από πολλαπλασιασμούς με το 2, οι οποίοι δεν είναι βέβαιο ότι θα σταματήσουν. Επομένως, αν οι πολλαπλασιασμοί δεν σταματήσουν εφαρμόζουμε την *επιθυμητή ακρίβεια*. Οι συντελεστές διαβάζονται με τη σειρά που παράγονται.

Έστω ο αριθμός 0.3

Αριθμός	Ακέραιο Γινόμενο πολ/μού με το 2	Κλασματικό γινόμενο	Συντελεστής
0.3	0	0.6	0
0.6	1	0.2	1
0.2	0	0.4	0
0.4	0	0.8	0
0.8	1	0.6	1

Με *ακρίβεια 5 bit*, 0.3= 0.01001 (οι συντελεστές γράφονται με τη σειρά που παράγονται)
Με ακρίβεια 3 bit, 0.3=0.010
Με *10 bit* 0100110011

0.625

Αριθμός	Ακέραιο Γινόμενο	Κλασματικό γινόμενο	Συντελεστής
0.625	1	0.25	1
0.25	0	0.5	0
0.5	1	0	1

(**ΣΤΑΜΑΤΟΥΜΕ**)

Άρα 0.625=0.101

Έστω ότι θέλουμε το 13.625.
Μετατρέπουμε χωριστά: 13= 1101
0.625= 0.101
1101.101 = 13.625

ΑΝΤΙΣΤΡΟΦΟ: Έστω ότι θέλουμε να βρούμε τον αριθμό 10101.110

Βάζουμε τις δυνάμεις του 2, ξεκινώντας από τη μηδενική αριστερά της υποδιαστολής και αυξάνοντας , ενώ δεξιά της υποδιαστολής μειώνουμε.

2^4 2^3 2^2 2^1 2^0 2^{-1} 2^{-2} 2^{-3}
1 0 1 0 1. 1 1 0

$1x2^4+1x2^2 + 1x2^0 +1x2^{-1}+1x2^{-2}$

$1x2^{-1} = 1/2^1 =1/2=0.5$
$1x2^{-2} = 1/2^2 =1/4=0.25$

Τελικά ο αριθμός είναι 21.75

lect1PG11

ΠΡΟΣΘΕΣΗ

Έστω 2 **μη προσημασμένοι** αριθμοί: 5 και 3

```
 101
+011
```

Είναι φανερό ότι η πρόσθεση δύο δυαδικών αριθμών γίνεται κατά στήλη όπως ακριβώς η δεκαδική πρόσθεση. Η διαφορά είναι ότι όταν το άθροισμα υπερβεί το 2 μεταφέρονται ΔΥΑΔΕΣ όπως αντίστοιχα στο δεκαδικό σύστημα μεταφέρονται δεκάδες.

ΔΕΚΑΔΙΚΗ πρόσθεση

```
25+
17
1        (αυτή η μονάδα που μεταφέρθηκε αντιστοιχεί σε δεκάδα)
___
42
```

ΤΕΣΣΕΡΙΣ περιπτώσεις πρόσθεσης κατά στήλες στο δυαδικό
0+0 = 0
0+1 = 1
1+0 = 1
1+1 = 0 και 1 δυάδα μεταφέρεται αριστερά.
1+1 (το αποτέλεσμα κάνει 2, στο δυαδικό σύστημα 2 είναι γραμμένο ως 10 δηλαδή 0 μονάδες και μία δυάδα)

```
    1 +
    1 =
 1  __
     0
```

ΕΠΕΚΤΑΣΕΙΣ: Τι θα συμβεί αν 2 στήλες έχουν μονάδα και έχει κατεβεί από την προηγούμενη στήλη μονάδα

1
1
1

Η λογική είναι ότι προσθέτουμε τις 2 επάνω δηλαδή 1+0 και 1 κρατούμενο και στη συνέχεια το αποτέλεσμα με την Τρίτη δηλαδή 0+1=1 και φυσικά ένα κρατούμενο

```
     1+
     1+
     1 =
_1_______
  1  1  = 3
```

```
  0   1    1     0  +
  0   1    1     0  =
  1   1
  ___________________

  1   1    0     0    =12
```

Έχουμε 4 στήλες, ξεκινάμε από δεξιά
1η στήλη: 0+0 =0
2η στήλη: 1+1 = 0 και μία τετράδα μεταφέρεται στην τρίτη στήλη.
3η στήλη 1+1+1: Από τους 2 επάνω άσσους το άθροισμα είναι 0 και 1 κρατούμενο το οποίο είναι μία ΟΚΤΑΔΑ που μεταφέρεται στην 4η στήλη. Στη συνέχεια στο αποτέλεσμα 0 προστίθεται η μονάδα κρατούμενου του προηγούμενου βήματος για να δώσει αποτέλεσμα 1
4η στήλη: 0+0+1=1

```
     1    0    0    1  +
     1    1    1    1  =
 1   1    1    1
 _____________________
 1   1    0    0    0
```

ΚΑΝΟΝΕΣ:
- Αν έχω 2 άσσους (συνολικά στους δύο αριθμούς και στο κρατούμενο) βάζω αποτέλεσμα 0 και 1 κρατούμενο
- Αν έχω 3 άσσους βάζω αποτέλεσμα 1 και 1 κρατούμενο
- Αν έχω έναν άσσο βάζω αποτέλεσμα 1 και 0 κρατούμενο
- Αν έχω μόνο μηδενικά, 0

_____ ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

```
    119 +
    592  =
    11    Κρατούμενα
_________
    711
```

Ξεκινώντας από δεξιά 9+2 =11 (Επειδή 11 δεν υπάρχει ως βασικό ψηφίο του δεκαδικού συστήματος, γράφουμε τον άσσο στη θέση των μονάδων και η άλλη μονάδα του 11, η οποία αντιστοιχεί σε 10άδα θα μεταφερθεί ως κρατούμενο). Αντίστοιχα, στη δεύτερη στήλη 1+9+1 =11, γράφουμε τη μονάδα και η μονάδα που λαμβάνουμε ως κρατούμενο μεταφέρεται στη στήλη των εκατοντάδων).

ΔΙΑΦΟΡΕΣ με το δυαδικό σύστημα:
1) Τα τυχόν κρατούμενα που προκύπτουν αντιστοιχούν σε δυάδες, τετράδες, οκτάδες……
2) Υπάρχουν ΜΟΝΟ 5 πιθανές περιπτώσεις όσον αφορά την πρόσθεση σε στήλες. Οι περιπτώσεις είναι οι εξής:

0+0 = 0
0+1 = 1 και 0 κρατούμενο
1+0 = 1 και 0 κρατούμενο
1+1 = 0 και 1 κρατούμενο

```
    1 +
    1 =
   1___
   10
```

1+1 =2 όμως στο δυαδικό σύστημα ΔΕΝ υπάρχει βασικός αριθμός 2 =10.
5+5 = 10

ΠΕΜΠΤΗ περίπτωση, να εξετάσουμε τις 4 παραπάνω περιπτώσεις με ύπαρξη κρατούμενου.

0+0 + 0 κρατούμενο: 0 και 0 κρατούμενα στην επόμενη στήλη (σε μία στήλη οι προστιθέμενοι αριθμοί να έχουν τιμή Bit =0 ενώ από την προηγούμενη στήλη, δηλαδή από την προηγούμενη πρόσθεση να έχουμε επίσης κρατούμενο 0)
Πχ
```
  100 +
    100 =
```

0 + 1 + 0: αποτέλεσμα 1 και κρατούμενο 0
0 + 1 + 1: αποτέλεσμα 0 και κρατούμενο 1

1 + 0 + 0 = αποτέλεσμα 1 και κρατούμενο 0
1 + 0 + 1 = αποτέλεσμα 0, και κρατούμενο 1

1 + 1 + 0 = αποτέλεσμα 0 και κρατούμενο 1
1 + 1 + 1 = (1+1+1=3). Προσθέτουμε τους 2 πρώτους άσσους άρα έχουμε αποτέλεσμα 0 και κρατούμενο 1. Ο τρίτος άσσος προστίθεται στο αποτέλεσμα και τελικά έχουμε αποτέλεσμα 1 κρατούμενο 1

ΓΕΝΙΚΟΣ ΚΑΝΟΝΑΣ: Όταν προστίθενται 3 bit (τα bit των αριθμών συν το κρατούμενο από προηγούμενο βήμα), τότε:
- Αν το πλήθος των άσσων είναι άρτιο τότε το αποτέλεσμα είναι 0 αν είναι περιττό είναι 1 (αν η στήλη περιέχει έναν ή τρεις άσσους γράφουμε 1 αλλιώς 0 στο αποτέλεσμα)
- Αν το πλήθος των άσσων στη στήλη είναι >= 2 τότε το κρατούμενο είναι 1 αλλιώς είναι 0.

ΠΑΡΑΔΕΙΓΜΑ: Να προστεθούν οι αριθμοί 15+12.

ΛΥΣΗ
15 =1111
12 =1100

```
            1111+
            1100
           110
         _______________
           11011   = 16+8+2+1 =27
```

22+18: 10110 + 10010

```
          1 0 1 1 0  +
          1 0 0 1 0 =
        1 0 1 1 0
_____________________
        1 0 1 0 0 0   = 32 +8=40
```

128+128

```
           10000000+
           10000000 =


_________________________
           100000000 (ΔΕΝ χωράει σε ένα byte, άρα χρειαζόμαστε 2ο)
```

```
  00000001| 00000000
```

```
  010100 = 20
  101100 =-20
```

ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ

```
              137 +
              128 =

           ______1_____
              265
```

Το κρατούμενο που μεταφέρεται σε κάθε στήλη μετά από μία πρόσθεση αντιστοιχεί σε ένα βάρος
Στο παράδειγμα, η μία μονάδα που μεταφέρθηκε στη δεύτερη στήλη αντιστοιχεί σε δεκάδα επειδή 7+8 = 15 δηλαδή 5 μονάδες και μία δεκάδα

Αντίστοιχα στο δυαδικό σύστημα τυχόν κρατούμενα αντιστοιχούν σε δυάδες, τετράδες κ.ο.κ

ΠΕΡΙΠΤΩΣΕΙΣ ΠΡΟΣΘΕΣΗΣ bit: όταν προσθέτουμε τα bit δύο αριθμών Α

και Β, η πρόσθεση γίνεται κατά στήλη. Σε κάθε στήλη υπάρχουν Ai, Bi Ci

Δηλαδή σε κάθε στήλη υπάρχουν ένα bit του αριθμού Α, ένα του αριθμού Β, και ένα κρατούμενο από την αμέσως προηγούμενη στήλη.

A	**B**	**C**	**Άθροισμα**	**Νέο κρατούμενο**
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**1**	**0**
0	**1**	**0**	**1**	**0**
0	**1**	**1**	**0**	**1**
1	**0**	**0**	**1**	**0**
1	**0**	**1**	**0**	**1**
1	**1**	**0**	**0**	**1**
1	**1**	**1**	**1**	**1**

(1+1=2. το 2 στο δυαδικό σύστημα είναι 10)

Το 10 στο δεκαδικό γράφεται όπως το 2 στο δυαδικό.
$(10)_2=2$
$(10)_{10}=10$
$(10)_8=8$
$(10)_{16}=16$

```
    1
    1
    1
   1
_________
   11
```

Λεκτικά: **1+1+1 = (1+1)+1 = (0)+1 =1**
Από την παρένθεση προκύπτει και ένα κρατούμενο που μεταφέρεται στην επόμενη στήλη. Τελικά αποτέλεσμα 11=3

Έστω ότι θέλουμε να προσθέσουμε τους αριθμούς 15 + 21

```
 15=  01111
 21 = 10101

       01111 +
       10101 =
      11111
_________________
      100100 =36
```

Σε κάθε στήλη προστίθενται τα 2 bit των αριθμών + ένα κρατούμενο που έχει προκύψει από την αμέσως προηγούμενη (δεξιά) στήλη.

36+ 28 : 100100 + 011100

100100+
011100 =
1111

1000000 = 64

Όταν προσθέτουμε 2 bit των A, B συν κρατούμενο: Το άθροισμα είναι 1 όταν ανάμεσα στα 3 bit υπάρχει περιττό πλήθος μονάδων (μία ή τρεις). Είναι 0 όταν υπάρχει άρτιο πλήθος μονάδων. Το νέο κρατούμενο είναι 1 αν ανάμεσα στα 3 Bit υπάρχουν 2 ή περισσότερες μονάδες δηλαδή αν το άθροισμα είναι πάνω από 1.

lect2PG1

Συμπληρώματα αριθμών

Συμπλήρωμα (complement) ενός αριθμού N είναι ένας άλλος αριθμός, ο οποίος συμπληρώνει τον N ως προς έναν αριθμό αναφοράς

- Ως προς βάση μείον ένα: r^n-N-1
- Ως προς βάση: r^n-N

Δεκαδικό σύστημα: N=975:
Χρειαζόμαστε, τη βάση: r=10
Χρειαζόμαστε και το n, το οποίο είναι το πλήθος ψηφίων του αριθμού N του οποίου ψάχνουμε το συμπλήρωμα, n=3
Εφαρμόζουμε τους τύπους:

Ως προς βάση μείον 1 (ως προς 9): 10^3-975-1= 24
Ως προς βάση (ως προς 10) : 10^3-975=25

Το συμπλήρωμα ως προς βάση είναι κατά 1 μεγαλύτερο του συμπληρώματος βάσης μείον 1
Συμβολισμοί: Σ_r, Σ_{r-1}

1) Συμπλήρωμα ως προς 9: Όταν αφαιρούμε r^n-1, αυτό που προκύπτει είναι n εννιάρια.
Στο παράδειγμα, είναι r=10, n=3, οπότε το r^n είναι ίσο με 10^3, δηλαδή μία μονάδα ακολουθούμενη από 3 μηδενικά, 1000. Συνεπώς αν αφαιρέσω 1, θα καταλήξω σε 3 εννιάρια. Άρα, στο συμπλήρωμα ως προς 9 ενός δεκαδικού αριθμού, αρκεί να αφαιρέσουμε όλα τα ψηφία του ζητούμενου αριθμού από 9.

Π.χ. για το 975 αφαιρούμε όλα τα ψηφία από 9: 999-975= 024

2) Συμπλήρωμα ως προς 10: Αφαιρούμε r^n-N, όπου r^n είναι μία μονάδα ακολουθούμενη από n μηδενικά. Όσο αφαιρείται το N από r^n, αυτό που συμβαίνει είναι ότι το πρώτο ψηφίο του N από δεξιά αφαιρείται από 10 (δηλαδή από την βάση), ενώ τα άλλα από 9 (βάση μείον 1)

```
    1000
-    975
      11
_____________
      025
```

Π.χ. το συμπλήρωμα του 2768, αφαιρούμε το 8 από 10 και όλα τα άλλα από 9: 7232

ΔΥΑΔΙΚΟ ΣΥΣΤΗΜΑ:
0-0:0 και δεν έχουμε δανειζόμενο
0-1:1 και ένα δανειζόμενο (αποτέλεσμα 11) 0-1=-1 (στο δυαδικό σύστημα, το -1 = 11, με 2 bit)
1-0: 1 και δεν έχουμε δανειζόμενο
1-1: 0 και δεν έχουμε δανειζόμενο.

Έχουμε 2 συμπληρώματα, ως προς 1 (συμπλήρωμα βάσης μείον 1, βάση r=2) και ως προς 2 (συμπλήρωμα βάσης).

Συμπλήρωμα βάσης μείον 1: Ο τύπος εξακολουθεί να είναι r^n-N-1 = (r^n-1)-N

Έστω ότι N=10101, n=5. Γνωρίζουμε ότι $r^n=2^5=32$ και επίσης ότι 32=100000.
Αν αφαιρέσουμε 32-1 στο δυαδικό έχουμε:

```
    100000 –
    000001 =
    11111
_____________
    011111 = 31
```

Για να βρω την ποσότητα r^n-1 ουσιαστικά γράφω n μονάδες οπότε το Σ_1 ενός αριθμού N προκύπτει αν αφαιρέσω όλα τα ψηφία του N από 1. Στο παράδειγμα

Σ_1(10101) =01010. Λόγω του γεγονότος ότι r^n-1 είναι n μονάδες, όταν πάμε να βρούμε το Σ_1 ενός δυαδικού αριθμού N, ουσιαστικά όλα τα bit του N αφαιρούνται από 1 άρα απλώς αντιστρέφονται.

Σ_1(1000001) = 0111110
Σ_1(1000000) = 0111111
Σ_1(0101010) = 1010101

ΚΑΝΟΝΑΣ εύρεσης Σ_1(N): Αντιστρέφουμε τα bit του N

ΣΥΜΠΛΗΡΩΜΑ ΒΑΣΗΣ (συμπλήρωμα ως προς 2)
Έχουμε να υπολογίσουμε την ποσότητα r^n-N, Αυτό σημαίνει ότι από την ποσότητα που γράφεται ως μία μονάδα και n μηδενικά θα πρέπει να αφαιρέσουμε το N. Αυτό σημαίνει τα εξής:

Αν το N έχει στο δεξί μέρος μηδενικά, τότε αυτά τα μηδενικά (όσα είναι) θα παραμείνουν και στο συμπλήρωμα

Έστω ότι N=1001000

10000000 –
1001000 =
1111
_____
0111000

2) Τι θα συμβεί όταν συναντήσουμε τον πρώτο άσσο από δεξιά;;;; Ο πρώτος άσσος αφαιρείται από 0 και δίνει αποτέλεσμα 1 και ένα δανειζόμενο. Άρα, ο πρώτος άσσος από δεξιά παραμένει άσσος.

3) Το δανειζόμενο που κατέβηκε λόγω του πρώτου άσσου από δεξιά, θα οδηγήσει στο να κατεβαίνουν δανειζόμενα σε κάθε αφαίρεση από εκείνο το σημείο. Άρα, όλα τα bit τα οποία ακολουθούν τον πρώτο άσσο από δεξιά αντιστράφηκαν (έντονα, υπογραμμισμένα, πλαγιαστά)

ΚΑΝΟΝΑΣ Σ_2: Τυχόν μηδενικά στο δεξί μέρος παραμένουν 0, ο πρώτος άσσος από δεξιά παραμένει 1 και όλα τα άλλα αντιστρέφονται.

Παραδείγματα: 10101010 = 01010110

11110000 = 00010000

11110001 = 00001111

10000000 = ίδιος αριθμός 10000000 (όπως ακριβώς το Σ_{10} του 0 είναι το 10)

Έστω ότι προσθέτω δύο συμπληρωματικούς (ως προς 2) αριθμούς

10010011 +
01101101 =
_____
100000000

_____ ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ _____

Δεκαδικό σύστημα. Βάση είναι το 10, βάση μείον 1 είναι το 9.
Δύο συμπληρώματα: Συμπλήρωμα 9 , συμπλήρωμα 10.

Συμπλήρωμα 9: r^n-N-1

N= ο αριθμός του οποίου ψάχνουμε το συμπλήρωμα, έστω 542.
n = το πλήθος ψηφίων του N, στο παράδειγμα 3. Άρα r^n=1000
Σ_9(542)= 1000-542-1 = 457
r^n-N-1= (r^n-1)-N. Η ποσότητα (r^n-1) είναι μία μονάδα και n μηδενικά μείον 1, άρα αυτό σημαίνει ότι αν κάνω την αφαίρεση θα πάρω n εννιάρια. **Άρα για να βρω το συμπλήρωμα βάσης μείον 1 στο δεκαδικό αφαιρώ όλα τα ψηφία του αριθμού N από 9.** Π.χ. 6898. 3101.
Άρα, όταν θα πάμε να κάνουμε την ίδια δουλειά στο δυαδικό σύστημα, όλα τα ψηφία του δυαδικού αριθμού N, θα αφαιρούνται από ΜΟΝΑΔΑ.

Πχ ο αριθμός 1000 του δυαδικού συστήματος είναι το 8 άρα 8-1 =7=111

Συμπλήρωμα βάσης

Έστω πάλι ο αριθμός 542. Το συμπλήρωμα 10, είναι r^n-N.

```
 1000 -
  542
 111

_____
  458
```

Εδώ το Σ_{10} είναι κατά 1 μεγαλύτερο από το Σ_9 και προκύπτει **αν το πρώτο ψηφίο του Ν από δεξιά αφαιρεθεί από τη βάση 10 και τα άλλα από την βάση μείον 1 δηλαδή 9.**

Π.χ. Σ_{10}(85435) = 14565 (το 5 από δεξιά αφαιρείται από 10 και όλα τα άλλα από 9).
Αντίστοιχα, **στο δυαδικό σύστημα, το πρώτο ψηφίο του Ν αφαιρείται από 2 και τα άλλα από 1.**

ΣΥΜΠΛΗΡΩΜΑΤΑ δυαδικών αριθμών

Σ_1(N). Έστω ότι ο αριθμός Ν αποτελείται από 5 bit, πχ 10101, τότε $r^n=2^5=32=100000$. Άρα 32-1=31=11111.
Συνεπώς, για να βρούμε το Σ_1(10101), αφαιρούμε όλα τα bit του από άσσους.

11111-10101 = 01010 (δηλαδή τα bit του αριθμού Ν έχουν αντιστραφεί)

ΚΑΝΟΝΑΣ εύρεσης Σ_1(N): όλα τα bit του Ν αντιστρέφονται
Πχ:
Σ_1(100001) = 011110
Σ_1(010100) = 101011
Σ_1(111111) = 000000

Έστω ότι προσπαθούμε να χρησιμοποιήσουμε την αναπαράσταση Σ_1 για να αναπαραστήσουμε αρνητικούς αριθμούς.
+5 = **0**0000101, όπου το αριστερότερο bit που είναι 0 υποδηλώνει το θετικό πρόσημα. Το Σ1(00000101) = 11111010.

Τότε ο αριθμός **1**1111010 είναι το -5;;;;;;; Αν ναι, τότε +5 + (-5)=0

```
Άρα    00000101 +
       11111010  =

_________________
       11111111
```

Αν υποθέσω ότι αυτή η αναπαράσταση 11111111 με κάποιον τρόπο είναι το 0, τότε τι είναι αυτό: 00000000 (Το Σ1 δημιουργεί 2

αναπαραστάσεις του μηδενός και για αυτό δεν χρησιμοποιείται για την αποθήκευση αρνητικών αριθμών).

Συμπλήρωμα Βάσης: Έστω ότι N=10101000

r^n =100000000

Άρα το Σ_2(10101000) είναι

```
 r^n                  100000000 -
 N                     10101000 =
               _______11111_________________
                     01011000
```

1) Εφόσον το N έχει δεξιά bit ίσα με το 0, αυτά θα παραμείνουν 0
2) Ο πρώτος άσσος από δεξιά θα παραμείνει άσσος αφού αφαιρείται από 2 (στο δυαδικό 0-1 κάνει 1 και ένα δανειζόμενο)

```
  10-
   8 =
 _1__
  02

 0-
 1
1___
11 (το 11 στο δυαδικό είναι το -1)
```

3) Όλα τα ψηφία αριστερά του πρώτου άσσου που θα βρεθεί αντιστρέφονται

Παραδείγματα:
Σ_2 (1010110) = 0101010
Σ_2(10000000) = 10000000
Σ_2(11100011) = 00011101

ΚΑΝΟΝΕΣ
1. Ελέγχω τυχόν μηδενικά στο δεξί μέρος του N και τα αφήνω 0
2. Ο πρώτος άσσος από δεξιά μένει άσσος
3. Όλα τα άλλα bit μετά τον πρώτο άσσο αντιστρέφονται

_____ ΤΜΗΜΑ ΔΕΥΤΕΡΑΣ _____

ΔΕΚΑΔΙΚΟ ΣΥΣΤΗΜΑ αρίθμησης.
Το οποίο σημαίνει ότι η βάση r=10, n= πλήθος ψηφίων του αριθμού N.
Θεωρούμε έναν αριθμό N = 356. Ο αριθμός N έχει τρία ψηφία, επομένως το r^n είναι 10^3=1000 (μία μονάδα και τρία (n=3) μηδενικά).

Άρα, το συμπλήρωμα δέκα (βάσης), το οποίο συμβολίζεται Σ_r, δηλαδή Σ_{10} στο δεκαδικό θα είναι 10^3- 356 = 1000 – 356 = 644. Δηλαδή το 644 δείχνει την απόσταση του αριθμού 356 από τον αριθμό 10^3. Αυτό σημαίνει ότι 644+356 = 10^3=1000. Αν από το 1000 αυτό αφαιρούσαμε την αριστερή μονάδα για να μείνει το αποτέλεσμα της πράξης 644+356 τριψήφιο, τότε αυτό το αποτέλεσμα θα ήταν ΜΗΔΕΝ. Επειδή το hardware είναι σχεδιασμένο (θα δούμε πως) για να δουλεύει έτσι την πρόσθεση, το συμπλήρωμα βάσης ενός αριθμού στο δυαδικό σύστημα θα δούμε ότι αναπαριστά τον αντίθετό του. Δηλαδή ένας δυαδικός αριθμός όταν προστεθεί με το συμπλήρωμά του ως προς 2, Σ_2, θα δώσει αποτέλεσμα 0.

ΥΠΟΛΟΓΙΣΜΟΣ Συμπληρώματος βάσης στο δεκαδικό: Ουσιαστικά, το τελευταίο από δεξιά ψηφίο του N αφαιρείται από τη βάση (δηλαδή το 10) και όλα τα άλλα από 9

```
1000 -
 356 =
  11
_____
 644
```

Π.χ. το Σ_{10}(32758) = 67242 (το 8 αφαιρείται από 10 και όλα τα άλλα από 9)

ΣΥΜΠΛΗΡΩΜΑ ΒΑΣΗΣ ΜΕΙΟΝ 1 (Σ_9 για το δεκαδικό σύστημα)

Σ_9(N) =r^n-N-1 = (r^n-1)-N. Όμως r^n-1 = n εννιάρια. Π.χ., αν n=3, r^n=10^3=1000, άρα έχουμε 1000-1=999 (n=3 εννιάρια). Άρα μπορούμε να πούμε ότι το Σ_9(N) προκύπτει αν αφαιρέσουμε όλα τα ψηφία του N από 9.

Π.χ. Σ_9(84552) = 15447

ΓΙΑ οκταδικό σύστημα (ψηφία από 0-7):
Σ_8 (6745) =1033 (το 5 αφαιρείται από 8 τα άλλα από 7)
Σ_7 (6745) = 1032 (όλα αφαιρούνται από 7, ένα μικρότερο από το Σ_8)

_____ ΔΥΑΔΙΚΟ ΣΥΣΤΗΜΑ_____

Συμπλήρωμα βάσης μείον 1: Βάσει του κανόνα που αναπτύξαμε για το δεκαδικό σύστημα, όλα τα ψηφία του αριθμού N θα πρέπει να αφαιρεθούν από την βάση μείον 1 (Στο δεκαδικό, όλα τα ψηφία του N αφαιρέθηκαν από 9). **Εδώ, όλα τα ψηφία του N αφαιρούνται από 1.**

Έστω N=10101011. Τότε Σ_1(N) είναι 01010100. Τα bit αντιστράφηκαν.

ΚΑΝΟΝΑΣ Εύρεσης Συμπληρώματος Βάσης Μείον 1 ενός αριθμού:
Αντιστρέφουμε όλα τα bit

Ερώτηση: Μπορούμε να χρησιμοποιήσουμε το Σ_1 ενός αριθμού N για να αναπαραστήσουμε τον αντίστοιχο αρνητικό;
Απάντηση: Έστω N=+5. Γράφουμε το N σε ένα byte, δηλαδή
N= **0**0000101 (θετικό πρόσημο, το αριστερότερο bit είναι 0).
Ας υποθέσουμε ότι το -5 είναι το Σ_1(N), δηλαδή **1**1111010 (αρνητικό πρόσημο, άσσος το αριστερότερο bit).
Άρα πρέπει η πρόσθεση των 2 αριθμών πρέπει να δώσει αποτέλεσμα 0

```
00000101 +
11111010 =
_______________
11111111 (άρα αυτό είναι 0;;;;;;;)
```
Αν αυτό είναι 0, τότε αυτό τι είναι: 00000000;;;;;;;;

Το Σ_1 δημιουργεί 2 αναπαραστάσεις του μηδενός

Το Σ_2 (N) είναι κατά 1 μεγαλύτερο του Σ_1(N).

Αν λοιπόν προσθέσω τους αριθμούς -5+5+1 θα πάρω το αποτέλεσμα

```
 00000101 +
 11111010 =
__________________
 11111111 +
        1 =
        1
_________________________
100000000
```

Η μονάδα αριστερά είναι εκτός ορίων του byte (οι αριθμοί που προστέθηκαν είναι 8 Bit), επομένως (ΘΑ ΤΟ ΔΟΥΜΕ ΣΤΟΥΣ ΚΑΝΟΝΕΣ ΠΡΟΣΘΕΣΗΣ) αγνοείται και το τελικό αποτέλεσμα είναι ΜΗΔΕΝ 00000000. Δηλαδή αν προσθέσω **N+Σ_1(N) +1 =0**. Όμως **Σ_1(N)+1=Σ_2(N)**, δηλαδή **το Σ_2(N) μας δίνει τον αντίθετο αριθμό του N** (ΝΑ ΤΟ ΕΧΕΤΕ ΥΠΟΨΗ όταν δείξουμε τη σχεδίαση του αθροιστή-αφαιρέτη)

_____ Συμπλήρωμα 2 _____

Στο δεκαδικό σύστημα είπαμε ότι το ***πρώτο ψηφίο από δεξιά αφαιρείται από 10 και τα άλλα από 9***, δηλαδή το πρώτο ψηφίο αφαιρείται από τη βάση και τα άλλα από βάση μείον 1. Σ_2(N) = r^n-N. Έστω ότι N είναι 8 bit ένα byte, τότε r^n = 2^8 = 1 μονάδα και 8 μηδενικά. Έστω ότι N = 10101100

```
rⁿ   100000000 -
N     10101100 =
         1
_________________
      01010100
```

1ο σκέλος: Τυχόν μηδενικά στο δεξί μέρος του Ν παραμένουν 0 (αφαιρούνται από 0)
2ο σκέλος: Τι θα συμβεί με την πρώτη μονάδα του Ν που εμφανίζεται δεξιά; Πρέπει να εκτελεστεί μία πράξη 0-1. Η μονάδα αυτή παραμένει 1 (υπογραμμισμένη). Από την πρώτη μονάδα που βρίσκουμε και πηγαίνοντας αριστερά θα έχουμε διαρκώς δανειζόμενα, δηλαδή οι πράξεις θα είναι είτε 0-**0**-1 είτε 0-**1**-1. Με έντονα γράμματα σημειώνουμε το Bit του αριθμού Ν, τα άλλα 2 bit είναι το bit του 2^n που είναι ΠΑΝΤΑ 0 και το δανειζόμενο που είναι 1).

0-**0**-1=1
0-**1**-1 =0 ,
δηλαδή τα bit του Ν ΑΝΤΙΣΤΡΕΦΟΝΤΑΙ σε κάθε περίπτωση

3ο σκέλος: Μετά την πρώτη μονάδα, όλα τα bit του Ν αντιστρέφονται

```
 0 -
 1
_____
```
11 (δανειζόμαστε μία δυάδα και κάνουμε την πράξη 2-1= 1 και ένα δανειζόμενο). Δηλαδή 0-1=11 (το 11 = -1)

```
10-
18
____
02
```

ΠΩΣ γίνεται η πράξη

```
      0-
      1-
      1

   __1___
      0
```

Παίρνουμε τα 2 πρώτα 0-1 =1 (ΑΠΟΤΕΛΕΣΜΑ) και 1 δανειζόμενο. Στη συνέχεια από το ΑΠΟΤΕΛΕΣΜΑ αφαιρείται η Τρίτη μονάδα δίνοντας ΤΕΛΙΚΟ αποτέλεσμα 0

ΚΑΝΟΝΑΣ: Δεξιά μηδενικά παραμένουν μηδέν, η πρώτη μονάδα από δεξιά παραμένει 1, όλα τα άλλα αντιστρέφονται.

Παραδείγματα: (10100000) = 01100000

(01011111) = 10100001
(10000000) = 10000000 (ΣΗΜΑΝΤΙΚΟ!!!!!!)

lect2PG2

ΑΝΑΠΑΡΑΣΤΑΣΗ ΑΡΝΗΤΙΚΩΝ

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

Χρησιμοποιείται το Σ_2 γιατί έχει την ιδιότητα ότι αν προσθέσουμε 2 συμπληρωματικούς και αγνοήσουμε το αριστερότερο bit που θα είναι 1 (ΑΥΤΌ γίνεται στην πραγματικότητα όταν κάνουμε πρόσθεση), τότε το αποτέλεσμα που απομένει είναι 0.

Το Σ_1 αυτό δεν ισχύει: πχ. 11110000 +
00001111 =

11111111
(ΕΊΝΑΙ ΔΥΝΑΤΟΝ ΑΥΤΌ ΝΑ ΕΊΝΑΙ 0;;;;;;) τότε 00000000 =0;
(ΔΙΠΛΗ ΑΝΑΠΑΡΑΣΤΑΣΗ του 0)

Ωστόσο, 11111111 +
1 =

100000000 (το αποτέλεσμα που βρήκαμε όταν αθροίσαμε δύο αριθμούς συμπληρωματικούς ως προς 2)

Π.χ. ο αριθμός 5 σε 1 byte είναι 00000101
Το -5 είναι το Σ_2 του 5 δηλαδή 11111011

Αν κάνουμε την πράξη 5+ (-5): **1** 00000000. Ο άσσος είναι έξω από τα όρια του byte και *αγνοείται*. Το υπόλοιπο είναι 0

Ο αριθμός 55 = 00110111, άρα το -55 = 11001001. Αν προσθέσουμε 55+(-55) θα πάρουμε 1 00000000 =0

ΣΗΜΑΝΤΙΚΑ ΕΡΩΤΗΜΑΤΑ

Ποιος είναι ο μεγαλύτερος ΜΗ προσημασμένος αριθμός που χωράει σε 1 byte;
11111111 = **255**

Ποιος είναι ο μεγαλύτερος ΜΗ προσημασμένος αριθμός που χωράει σε 2 bytes;
11111111 11111111 = 2^{16}-1 = **65535**

Ποιος είναι ο μικρότερος ΜΗ προσημασμένος αριθμός που χωράει σε 1 byte;
00000000 = **0**

Ποιος είναι ο μικρότερος ΜΗ προσημασμένος αριθμός που χωράει σε 2 byte;
00000000 00000000

Ποιος είναι ο μεγαλύτερος προσημασμένος αριθμός που χωράει σε 1

byte;
01111111 = **+127** (7 μονάδες και το αριστερό 0 είναι το πρόσημο)

Ποιος είναι ο μεγαλύτερος προσημασμένος αριθμός που χωράει σε 2 byte;
01111111 11111111= **+32767** (15 μονάδες = 2^{15}-1 και το αριστερό 0 είναι το πρόσημο)

Ποιος είναι ο μικρότερος προσημασμένος αριθμός που χωράει σε 1 byte;
10000000. Αν πάρουμε το Σ_2 αυτού του αριθμού αυτό είναι το **128**. Άρα πρόκειται για το 128

Πράξη: **-128 +127**

```
     01111111
  +  10000000  =
_______________
     11111111 (ΠΟΙΟΣ αριθμός είναι οι 8 μονάδες;)
```

Για να βρούμε τον αριθμό, επειδή είναι *αρνητικός*, δεν μπορούμε να εφαρμόσουμε όσα ξέραμε με δυνάμεις του 2. Βρίσκουμε το
Σ_2(11111111) = 00000001.
Επειδή *00000001=1* άρα *11111111=-1*

Ποιος είναι ο μικρότερος προσημασμένος αριθμός που χωράει σε 2 byte;
10000000 00000000 = -2^{15}=**-32768**

10101010 + 01010110 = 1 00000000 (8 ψηφία, η αναπαράσταση είναι 1 άσσος και 8 μηδενικά)
975+25 = 1000 (έναν άσσο και 3 μηδενικά)

Έστω ότι προσθέτουμε -32768 + 32767

```
   1000000000000000 +
   0111111111111111 =
_____________________
   1111111111111111 =-1
```

Δηλαδή με 2 byte 16 άσσοι είναι -1
Σε 1 bytes 8 άσσοι είναι πάλι -1

Ποιος είναι ο αριθμός -256; 1 00000000

Το -256 απαιτεί 2 byte: 11111111 00000000

Για να καταλάβουμε ότι αυτό είναι το -256 βρίσκουμε το Σ_2:
0000000100000000 =256

Προσθέσαμε 2 αριθμούς συμπληρωματικούς ως προς 2 του ενός byte και βγάλαμε **1** 00000000 . Σε ένα Byte αυτό είναι 0. Για να γράψουμε την δύναμη 2^n θέλουμε 2^o Byte.

Unsigned n = 128
Signed n =-128

Έστω ότι η γλώσσα δεσμεύει
2 byte για unsigned. 0000000010000000
2 bytes για signed: ΙΔΙΑ 1111111110000000

_____ ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Δίνεται ο αριθμός +132. Να γραφτεί ο αριθμός σε κατάλληλο πλήθος byte. Να βρεθεί ο αριθμός -132. Να προστεθούν και να δείξετε ότι το αποτέλεσμα είναι 0. Να επαναλάβετε με τους αριθμούς +18, -18

128+4:

Από την αναπαράσταση *ΜΗ προσημασμένων* ξέρουμε ότι 132= 10000100. Αν ο αριθμός είναι προσημασμένος θέλουμε 2ο byte (παρότι θέλουμε μόνο ένα bit θα ξοδέψουμε ένα byte)

Με 4 bytes: 000000000 0000000 00000000 10000100 = +132

00000000 10000100 = +132
Δηλαδή, αν η γλώσσα προγραμματισμού δέσμευε 2 byte τότε το +132 θα ήταν **0**0000000 10000100

int k=132;
unsingned k =132;

Αν ο unsigned τύπος δέσμευε 2 byte η αναπαράσταση θα ήταν ίδια με τον signed. ΑΛΛΑ αν δέσμευε ένα θα ήταν 10000100. Δηλαδή σε unsinged, ο αριθμός 132 ΧΩΡΑΕΙ σε 1 byte ο signed ΌΧΙ

-132: είναι το Σ_2(00000000 10000100) = **1**1111111 01111100

ΠΩΣ μπορούμε να ξέρουμε ΕΜΕΙΣ (ο υπολογιστής ξέρει) ποιος αριθμός αναπαρίσταται ως **1**1111111 01111100

ΑΠΑΝΤΗΣΗ: Αν ο αριθμός συμβεί να είναι μη προσημασμένος, με τον τρόπο μετατροπής. **Αν όμως είναι προσημασμένος για να βρείτε ποιος είναι ο αριθμός θα πάρετε συμπλήρωμα ως προς 2 για να βρείτε τον αντίστοιχο θετικό**:

Σ_2(11111111 01111100) = 00000000 10000100.
Επειδή αυτός ο αριθμός είναι 132 ο αρνητικός είναι -132

ΠΡΟΣΘΕΣΗ

  00000000 10000**1**00 +
  11111111 01111**1**00
_____
1 00000000 00000000 (κάνοντας τις πράξεις βλέπουμε από το τρίτο Bit ξεκινώντας από δεξιά και μετά έχουμε συνεχώς κρατούμενα

και συνεχώς την πράξη 1+1=0 και 1 κρατούμενο)

ΑΠΟΤΕΛΕΣΜΑ : **1** 00000000 00000000 (αυτός συνολικά είναι ο αριθμός αναφοράς 2^{16}). Αν αγνοήσω τον αριστερό άσσο (ΚΑΤΙ ΤΟ ΟΠΟΙΟ ΣΥΜΒΑΙΝΕΙ ΣΤΗΝ ΠΡΑΞΗ ΤΗΣ ΠΡΟΣΘΕΣΗΣ), το αποτέλεσμα είναι 0.

Έστω ότι το σύστημά μας αποθηκεύει με 1 byte

+18 = **0**00 10010

-18 = **1**11 01110

_____
1 00000000

Με άλλα λόγια το Σ_2 έχει την ικανότητα, όταν προστεθούν 2 συμπληρωματικοί αριθμοί να δώσει αποτέλεσμα το r^n, από το οποίο, αν αφαιρεθεί η αριστερότερη μονάδα να απομένει 0. Άρα *οι 2 συμπληρωματικοί αριθμοί είναι αντίθετοι*.

Προσημασμένοι ενός byte:

Μεγαλύτερος 01111111 = +127
Μικρότερος 10000000 = -128 γιατί αν ψάξουμε το Σ_2(1000000) = 10000000 το ίδιο που σε θετική ποσότητα είναι 128

Το +128 ΔΕΝ χωράει σε 1 byte ενώ το -128 χωράει.
-128 +127:
10000000 +
01111111 =
_____
11111111
(ΠΟΙΟΣ είναι αυτός ο αριθμός; ΔΕΝ ξέρουμε πρέπει να βρούμε το Σ_2)

Είναι 000000001 άρα είναι το -1

Σε **2 byte** 10000000 00000000 = -2^{15}=**-32768**
Μεγαλύτερος 01111111 11111111 = **32767**.

_____ ΤΜΗΜΑ ΔΕΥΤΕΡΑΣ _____

Ποιος είναι ο αριθμός -28 (γραμμένος σε 1 byte);
Δίνεται η αναπαράσταση 10101011 σε προσημασμένη μορφή. Ποιος είναι αυτός ο αριθμός;
Δίνεται ο αριθμός 10000000. Να εκτελέσετε την πρόσθεση με τον αριθμό 01111111. Ποιο είναι το αποτέλεσμα; Ποιος είναι σε δεκαδική μορφή ο αριθμός 10000000; Ποιος είναι σε προσημασμένη μορφή ο αριθμός +128;;;

ΑΠΑΝΤΗΣΕΙΣ

1) Γράφουμε τον + 28 = **0**0011100 (το έντονο bit είναι το πρόσημο +). Επομένως, το -28 είναι το Σ_2(00011100), δηλαδή 11100100. Προσθέτοντας, έχουμε

```
   00011100 +
   11100100 =
_____________
  1 00000000
```

(ΠΑΝΤΑ θα παίρνουμε μηδενικά και μία μονάδα έξω από τα όρια των αριθμών)

Ο αριθμός 11100100 είναι το συμπλήρωμα του 00011100 ως προς την αναφορά 2^8. Αν από το 2^8 αγνοήσουμε τον αριστερό άσσο παίρνουμε 0

2) 10101011. Επειδή είναι ΠΡΟΣΗΜΑΣΜΕΝΟΣ ΑΡΝΗΤΙΚΟΣ δεν μπορούμε να εφαρμόσουμε τη μέθοδο που ξέρουμε. Βρίσκουμε το Σ_2 του αριθμού για να βρούμε τον αντίστοιχο θετικό και μετά βρίσκουμε τον αρνητικό.

Σ_2(10101011) = 01010101 = 1 + 4 + 16 + 64 = 85.
Άρα ο ζητούμενος αριθμός είναι το -85

```
3)   10000000 +
     01111111  =
_______________
     11111111 (το αποτέλεσμα αυτό είναι ένας αρνητικός αριθμός)
```

Άρα, ποιος είναι ο αντίστοιχος θετικός;
Είναι το Σ_2(11111111) = 00000001 = 1.
Άρα, ο αριθμός 11111111 = -1
Από την άλλη ο αριθμός **01111111= 2^7-1=127**, κάτι που σημαίνει ότι ο αριθμός **10000000 = -128** (ο αριθμός προστέθηκε στο +127 και έδωσε αποτέλεσμα -1).

ΚΑΙ ΑΦΟΥ είναι ΕΤΣΙ, ποιος αριθμός είναι το +128; **0** 10000000

00000000 10000000 (άρα χρειαζόμαστε 2^o byte).
Ο αριθμός +128 ΔΕΝ μπορεί να αναπαρασταθεί με 1 byte.
Ο -128 "χωράει" σε ένα byte.

Αν θέλουμε το -128 σε 2 byte βρίσκουμε το
Σ_2(**0**0000000 10000000) = **1**1111111 10000000 = -128 με 2 byte

Ποιος είναι ο *μεγαλύτερος unsigned integer* που χωράει σε 1 byte;

11111111 = 2^8-1=255.

Με 2 bytes; 11111111 11111111 = 2^{16}-1 = 65.535

2) Ποιος είναι ο *μεγαλύτερος signed integer* που χωράει σε 1 byte

01111111 = 2^7-1=**127**

Με 2 bytes; 0111111 11111111 = **2^{15}-1** = 32767

3) Ποιος είναι ο *μικρότερος unsigned integer* που χωράει σε 1 byte

00000000=**0**. Ομοίως και σε 2 bytes

4) Ποιος είναι ο *μικρότερος signed integer* που χωράει σε 1 byte = 8 bit άρα 256 συνδυασμοί των bit;

10000000 =**-128**.
Σε 1 byte, μπορώ να έχω 256 διαφορετικούς ακέραιους (2^8=256).
Οι 128 είναι θετικοί (0-127),
οι 128 είναι αρνητικοί από -1 ως -128
Όπου -1=11111111
-128 = 10000000

Σε 2 bytes: 10000000 00000000 = -2^{15}=-32768

lect3PG1

ΠΡΑΞΕΙΣ ΜΕ ΠΡΟΣΗΜΑΣΜΕΝΟΥΣ

_____ΤΜΗΜΑ ΔΕΥΤΕΡΑΣ_____

Σε κάθε περίπτωση, γίνεται πρόσθεση.
ΚΑΝΟΝΑΣ: Οι θετικοί αριθμοί γράφονται ως έχουν. **Οι αρνητικοί, γράφονται σε μορφή Σ2 και γίνεται πρόσθεση**. Αυτό μας δίνει το πλεονέκτημα κατά την υλοποίηση του hardware να χρησιμοποιήσουμε *ΈΝΑ ΜΟΝΟ* κύκλωμα, το οποίο κάνει πρόσθεση και αφαίρεση.

ΣΗΜΑΝΤΙΚΟ: ΝΑ ΚΑΤΑΛΑΒΟΥΜΕ ΣΕ ΠΟΙΕΣ ΠΕΡΙΠΤΩΣΕΙΣ ΕΧΟΥΜΕ ΥΠΕΡΧΕΙΛΙΣΗ και σε ποιες όχι. Overflow.

Στα 4 παραδείγματα θα χρησιμοποιήσουμε προσημασμένους αριθμούς (A=96 και B=50) ενός byte.

ΕΡΩΤΗΣΗ:
Με δεδομένο ότι με 1 byte μπορούμε να χωρέσουμε τους αριθμούς από **-128 ως 127**, σε ποιες από τις παραπάνω 4 πράξεις μπορεί το αποτέλεσμα να φύγει έξω από το διάστημα **[-128…127]**;

A+B: 2 θετικοί αριθμοί στο [-128…127] μπορούν να μας βγάλουν αποτέλεσμα ε*κτός ΔΕΞΙΟΥ* ορίου (>127)
-A-B = -(A+B), μπορούν να μας βγάλουν αποτέλεσμα *εκτός ΑΡΙΣΤΕΡΟΥ ορίου* (<-128)
Οι 2 αυτές περιπτώσεις μπορούν να οδηγήσουν σε **υπερχείλιση ΠΡΟΣΗΜΟΥ**.

Στις άλλες 2 περιπτώσεις, δεν μπορούμε να βγούμε εκτός διαστήματος. ΔΕΝ υπάρχει υπερχείλιση
Αντίστοιχα θα ισχύουν και με περισσότερα byte (το διάστημα θα είναι μεγαλύτερο).

ΠΩΣ ελέγχεται η υπερχείλιση; ΑΠΟ τα 2 τελευταία ΚΡΑΤΟΥΜΕΝΑ των πράξεων.
Αν τα 2 τελευταία κρατούμενα:
- είναι ίσα (είτε 0 είτε 1) ΔΕΝ ΕΧΟΥΜΕ υπερχείλιση προσήμου,
- αν δεν είναι ίσα (το προτελευταίο κρατούμενο να είναι 0 και το τελευταίο να είναι 1) τότε έχουμε υπερχείλιση.

ΤΜΗΜΑ ΠΕΜΠΤΗΣ

Θα χρησιμοποιήσουμε αριθμούς ενός byte αλλά σε ανάλογα συμπεράσματα καταλήγουμε και για 2 ή περισσότερα byte.
96-50
50-96
96+50
-96-50

Οι προσημασμένοι αριθμοί ενός byte Δ= [−128… 127]
Α, Β εντός του Δ. Σε ποιες περιπτώσεις μπορεί το αποτέλεσμα να «υπερχειλίσει» δηλαδή να βγει εκτός του Δ. Στις περιπτώσεις 3, 4.
ΥΠΕΡΧΕΙΛΙΣΗ (Overflow)
Έχω δηλώσει τους αριθμούς Α και Β στο πρόγραμμά μου ως integer και έστω ότι η γλώσσα προγραμματισμού αποθηκεύει τους ακεραίους σε 1 byte.
S= A+B, το S θα αποθηκευτεί σε 1 byte.
96+50=146 (ΕΚΤΟΣ ΟΡΙΟΥ)
96+2=98

ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ

Οι Α και Β είναι ενός byte
Οι Α και Β ανήκουν στο Δ=[-128..127]
-A-B =-(A+B)

lect3PG2

A-B=96-50

96= 01100000
50= 00110010
-96= 10100000
-50= 11001110

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	ΘΕΣΕΙΣ
	0	1	1	0	0	0	0	0	Ai
	1	1	0	0	1	1	1	0	Bi
	1	0	0	0	0	0	0	0	Ci
1	0	0	1	0	1	1	1	0	Si

Θέσεις: Δυνάμεις του 2.
Ai είναι τα bit του αριθμού A
Bi είναι τα bit του αριθμού B
Ci είναι τα κρατούμενα που προκύπτουν από πράξη στην αμέσως προηγούμενη στήλη

C0 είναι το αρχικό κρατούμενο
C1 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A0+B0
C2 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A1+B1
C3 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A2+B2

Με **κόκκινα γράμματα** είναι το C8, το οποίο είναι το κρατούμενο που προκύπτει από την πρόσθεση A7+B7
Για τους ελέγχους υπερχείλισης θα εξετάζουμε τα **C7, C8**

1η περίπτωση: A-B με A>B: **Προσθέτουμε στο Α το Σ2 του Β**, δηλαδή στο +96 το -50. Το αποτέλεσμα θα έχει μία μονάδα στο C8 (κόκκινο) η οποία *αγνοείται*. Το υπόλοιπο είναι το σωστό αποτέλεσμα.
00101110 = +46
 ΥΠΑΡΧΕΙ υπερχείλιση; ΟΧΙ επειδή **C7=C8=1(ΠΑΝΤΑ όταν A>B)**. Όταν το κύκλωμα αθροιστή ελέγξει τα 2 τελευταία κρατούμενα και δει ότι είναι ίσα, θα στείλει ένα σήμα ότι το αποτέλεσμα είναι σωστό.
ΑΠΟΔΕΙΞΗ ΟΡΘΟΤΗΤΑΣ: Το πρώτο bit από αριστερά είναι 0, θετικό όπως θα έπρεπε να είναι.
Όλες οι πράξεις αυτής της μορφής υπακούν στους ίδιους κανόνες και συμπεράσματα.

ΠΑΡΑΤΗΡΗΣΗ: ΓΙΑ 2 byte ελέγχετε τα κρατούμενα C15, C16

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

Θέσεις: Δυνάμεις του 2.
Ai είναι τα bit του αριθμού A
Bi είναι τα bit του αριθμού B
Ci είναι τα κρατούμενα που προκύπτουν από πράξη στην αμέσως προηγούμενη στήλη

C0 είναι το αρχικό κρατούμενο
C1 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A0+B0
C2 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A1+B1
C3 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A2+B2

Και τελικά το C8 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A7+B7.
Το C8 θα είναι κόκκινο και θα το κατεβάζουμε στο αριστερότερο bit

του S

ΠΑΝΤΑ ΙΣΧΥΕΙ ΤΟ ΕΞΗΣ
Αν ο αριθμός που λαμβάνει μέρος στο άθροισμα είναι θετικός, γράφεται στην κανονική του μορφή.
Αν είναι αρνητικός, λαμβάνουμε το Σ2
Σε κάθε περίπτωση ΠΡΟΣΘΕΤΟΥΜΕ.
Π.χ. 96-50. Προσθέτουμε στο +96 το Σ2 του +50

Για να κάνουμε έλεγχο υπερχείλισης, θα ελέγχουμε τα κρατούμενα C7,C8.

A-B είναι σίγουρο ότι με A>B θα πρέπει να δώσει θετικό αποτέλεσμα.

ΑΝΑΛΥΣΗ Είναι προφανές ότι επειδή έχουμε προσθέσει τους αριθμούς, ότι θα έχουμε:

S7= A7+B7 + C7 = 0+1 +C7, Δηλαδή S7= 1 +C7.

Αν το C7=0 τότε το αποτέλεσμα S7 θα είναι ίσο με 1 (ΓΙΝΕΤΑΙ;;;;;;;;;)
Άρα, για να βγει το αποτέλεσμα θετικό, πρέπει C7=1
Αν το C7=1, τότε S8=1, γιατί στην αριστερότερη στήλη εκτελείται η πράξη

S7= A7 + B7 + C7 = 1+0+1. Άρα S7=0 και C8=1.

ΣΥΜΠΕΡΑΣΜΑ
Άρα, αν το αποτέλεσμα είναι θετικό, τότε θα είναι C7=1=C8.
Αν έχουμε A-B και A>B, τότε C7=C8=1.
Σε αυτή την περίπτωση, όπου C7=C8=1, το C8 αγνοείται, απορρίπτεται και τα υπόλοιπα 8 bit δείχνουν κανονικά το αποτέλεσμα.

00101110 = 2+4+8+32=+46

01100000-
00110010 =
_____
00101110

ΤΕΛΙΚΗ ΠΑΡΑΤΗΡΗΣΗ: Αν έχουμε αριθμούς 2 byte ελέγχουμε τα κρατούμενα C15, C16 (αριστερότερα)

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Αν έχουμε την πράξη ανάμεσα σε 2 αριθμούς A,B, όπου κάποιος ή και οι δύο εμφανίζονται σε αρνητική μορφή, τότε για να εκτελέσουμε την πράξη παίρνουμε το Συμπλήρωμά τους. Σε κάθε περίπτωση η πράξη ανάγεται σε πρόσθεση

96-50 = 96 + Σ2(50)

Θέσεις: Δυνάμεις του 2.
Ai είναι τα bit του αριθμού A
Bi είναι τα bit του αριθμού B
Ci είναι τα κρατούμενα που προκύπτουν από πράξη στην αμέσως προηγούμενη στήλη
C0 είναι το αρχικό κρατούμενο
C1 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A0+B0
C2 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A1+B1
C3 είναι το κρατούμενο που προκύπτει από την πρόσθεση των bit A2+B2

Με κόκκινα γράμματα είναι το C8, το οποίο είναι το κρατούμενο που προκύπτει από την πρόσθεση A7+B7+C7
Για τους ελέγχους υπερχείλισης θα εξετάζουμε τα C7, C8

Αν κάνουμε την ίδια δουλειά για 2 byte, μας ενδιαφέρουν τα τελευταία 2 κρατούμενα τα οποία θα είναι στις θέσεις C15, C16

```
  787+
  532
_______
 1319
```

ΑΝΑΛΥΣΗ αποτελέσματος:

Το αποτέλεσμα που προέκυψε έχει ως πρόσημο το 0, δηλαδή είναι θετικό (Θέση S7).
Έχουμε προσθέσει έναν θετικό αριθμό A (A7=0) με έναν αρνητικό B (B7=1).
Για να είναι το αποτέλεσμα θετικό, πρέπει το S7=0.
Όμως S7= A7+B7+C7 = 0+1+C7. Άρα S7=1+C7.
Για να είναι S7=0 και το αποτέλεσμα να είναι θετικό πρέπει C7=1.
Εφόσον C7=1, η πράξη S7= A7+B7+C7 θα δώσει αποτέλεσμα S7=0 και C8=1

ΣΥΜΠΕΡΑΣΜΑ: Αν C7=C8=1, τότε το αποτέλεσμα της αφαίρεσης είναι θετικός αριθμός, και το C8 αγνοείται. Τα υπόλοιπα Bit του S σχηματίζουν το τελικό αποτέλεσμα.
Στο παράδειγμά μας το S= 00101110 = 2+4+8+32=46.

```
    01100000 –
    00110010 =
 ____________
    00101110
```

```
A =  00000000 01100000
B =  11111111 11001110
```

lect3PG3

-A+B=-96+50

96= 01100000
50= 00110010
-96= 10100000
-50= 11001110

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	ΘΕΣΕΙΣ
	1	0	1	0	0	0	0	0	Ai
	0	0	1	1	0	0	1	0	Bi
	0	1	0	0	0	0	0	0	Ci
0	1	1	0	1	0	0	1	0	Si

_____ΤΜΗΜΑ ΔΕΥΤΕΡΑΣ_____

B-A: Επειδή B<A, το αποτέλεσμα θα είναι *αρνητικό*. **Τα κρατούμενα C7,C8 θα είναι ίσα μεταξύ τους με τιμή 0, δηλαδή δεν θα υπάρχει υπερχείλιση** και το αποτέλεσμα θα έχει άσσο στο αριστερότερο bit.
Επομένως, για να ελέγξουμε σε ποιον αριθμό αντιστοιχεί αυτό το αρνητικό αποτέλεσμα θα πρέπει να πάρουμε το Σ2.
Πήραμε το Σ2 του 96 και το προσθέσαμε στο +50
Το τελευταίο κρατούμενο C8 είναι 0 και αγνοείται (αφού είναι 0) και το υπόλοιπο είναι το σωστό αποτέλεσμα δηλαδή 11010010
ΠΟΙΟΣ αριθμός είναι το αποτέλεσμα; Παίρνω Σ_2 (11010010) =00101110.
Επειδή 00101110= 2+4+8+32=46. το 11010010=-46
ΔΕΝ ΕΧΕΙ ΥΠΕΡΧΕΙΛΙΣΗ επειδή **C7=C8=0** και το αποτέλεσμα είναι σωστό επειδή το αριστερότερο bit είναι 1.

Ο αθροιστής ελέγχει ότι C7=C8 =0 και στέλνει σήμα ορθότητας του αποτελέσματος.

ΤΕΛΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ:
- Πάντα, όταν τα 2 τελευταία κρατούμενα είναι 0, ο αριθμός που προκύπτει ως αποτέλεσμα θα είναι αρνητικός.
- Όταν τα 2 τελευταία κρατούμενα είναι 1, τότε ο αριθμός που προκύπτει ως αποτέλεσμα είναι θετικός.

ΑΝΑΛΥΣΗ ΤΟΥ ΑΠΟΤΕΛΕΣΜΑΤΟΣ: Στο συγκεκριμένο παράδειγμα έχουμε την πρόσθεση 2 αριθμών εκ των οποίων ένας είναι θετικός και ένας αρνητικός.
Άρα, ο ένας έχει αριστερότερο bit =1 και ο άλλος 0.

A7=1
B7=0.

Θέλουμε το αποτέλεσμα B-A να είναι αρνητικό, δεδομένου ότι σε απόλυτη τιμή το A >B (A=96, B=50). Άρα, το αποτέλεσμα S7=1
Όμως S7=A7+B7+C7 ή S7=1+0+C7. Άρα, για να είναι S7=1, πρέπει C7=0. Εφόσον C7=0, η πράξη A7+B7+C7 θα είναι 1+0+0 και θα δώσει κρατούμενο C8=0.
Άρα C7=C8=0

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

Εδώ έχουμε πάλι ότι προστίθεται ο θετικός B (B7=0) με τον αρνητικό A (Άρα A7=1)

S7=A7+B7+C7=> S7=1+0+C7
Άρα αν το C7 είναι ίσο με 1, τότε S7=0 και C8=1 (ΠΡΟΗΓΟΥΜΕΝΗ ΠΕΡΙΠΤΩΣΗ)

Για να είναι λοιπόν το αποτέλεσμα αρνητικό πρέπει C7=0, οπότε S7=1
Τότε, η πράξη είναι S7=A7+B7+C7=> S7=1+0+0= 1 και C8=0

ΣΥΜΠΕΡΑΣΜΑ: Αν C7=C8=0, τότε το αποτέλεσμα είναι αρνητικό και το κρατούμενο C8 αγνοείται.

Αποτέλεσμα: **1**1010010. Ο αριθμός αυτός είναι κάποιος αρνητικός. Για να βρούμε ποιος είναι βρίσκουμε το **Σ2** (11010010) = 00101110.

Και επειδή 00101110=+46, το αποτέλεσμά μας είναι -46

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Προσθέτουμε στο 50 το Σ2(96)
Προσθέτουμε τον αρνητικό αριθμό A με τον θετικό B. Αυτό σημαίνει ότι A7=1, B7=0.
S7=A7+B7+C7. Αν το αποτέλεσμα είναι αρνητικό, τότε πρέπει *C7=0*, έτσι ώστε *S7=1*

S7= 1+0+C7.
Αν όμως C7=0 τότε η πράξη της αριστερότερης στήλης θα δώσει S7=1+0+0=1 και S8=0.

Με άλλα λόγια όταν C7=C8=0, τότε το αποτέλεσμα της αφαίρεσης είναι αρνητικό και το C8 αγνοείται. Το υπόλοιπο S είναι το τελικό αποτέλεσμα

Στο παράδειγμα 11010010. ΠΟΙΟΣ αριθμός είναι; Σ2(00101110).
Επειδή 00101110=46 το αποτέλεσμά μας είναι -46.

Όταν C7=C8, τότε το αποτέλεσμα είναι σωστό και το C8 αγνοείται.
Ειδικότερα,
- Αν C7=C8=1, το αποτέλεσμα είναι θετικό
- Αν C7=C8=0, το αποτέλεσμα είναι αρνητικό

(Για πράξεις με αριθμούς ενός byte. Για παραπάνω, ελέγχουμε τα 2 αριστερότερα κρατούμενα, πχ για 2 bytes τα C15,C16)

lect3PG4

A+B=96+50

96= 01100000
50= 00110010
-96= 10100000
-50= 11001110

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	ΘΕΣΕΙΣ
	0	1	1	0	0	0	0	0	Ai
	0	0	1	1	0	0	1	0	Bi
	1	1	0	0	0	0	0	0	Ci
0	1	0	0	1	0	0	1	0	Si

Εδώ προσθέτουμε τους 2 θετικούς.

Από το αποτέλεσμα προκύπτει ότι 96+50 = κάποιος αρνητικός.
ΤΟ ΠΡΟΣΗΜΟ έχει υπερχειλίσει εκτός ορίων του αριθμού και είναι το κρατούμενο *C8*.

Αν γράψουμε τον αριθμό
0 10010010 = 146
αλλά το πρόσημο 0 είναι εκτός ορίων του αριθμού.
Ο αριθμός 146 **ΔΕΝ ΧΩΡΑΕΙ** σε 1 byte

ΑΝΑΛΥΣΗ αποτελέσματος:
Προσθέσαμε 2 θετικούς αριθμούς, άρα A7=B7=0.
Το αποτέλεσμα ήταν S7=1
Επειδή όμως S7=A7+B7+C7, σημαίνει ότι C7=1
Άρα, η πράξη δίνει S7= 0+0+1 δηλαδή S7=1 και C8=0

Δηλαδή C7=1, C8=0
(**ΠΟΛΎ ΣΗΜΑΝΤΙΚΟ:** το C8=0 είναι το θετικό πρόσημο που έχει υπερχειλίσει)

Επομένως, **όταν προστεθούν 2 θετικοί αριθμοί και συμβεί C7=1, C8=0, έχουμε υπερχείλιση του θετικού προσήμου.** Ο αθροιστής στέλνει σήμα στην CPU ότι υπάρχει σφάλμα αποτελέσματος. Ο προγραμματιστής θα το δει ως *overflow* μήνυμα.

A+B=50+50 ΔΕΝ έχουμε υπερχείλιση. Αν εκτελέσετε την πράξη, θα διαπιστώσετε ότι C7=C8

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

Έχουμε να προσθέσουμε 2 θετικούς Α, Β άρα *A7=B7=0*.
Αν συμβεί το αποτέλεσμα *S7=0*, τότε αυτό σημαίνει ότι *C7 = 0*

Αν όμως C7=1, τότε θα συμβεί το παράδοξο να προσθέσουμε 2 θετικούς και να λάβουμε αρνητικό αποτέλεσμα (ΌΠΩΣ ΣΤΟ ΠΑΡΑΔΕΙΓΜΑ).
Δηλαδή **S7=A7+B7+C7= 0+0+1=1** και C8=0

Σε αυτή την περίπτωση έχουμε **ΥΠΕΡΧΕΙΛΙΣΗ**!!!
Η υπερχείλιση αφορά στο ΠΡΟΣΗΜΟ. Αν C7= 1 και C8=0 τότε έχει συμβεί υπερχείλιση και το C8 αντιστοιχεί στο πρόσημο του αριθμού-αποτελέσματος, το οποίο έχει βγει έξω από τα όρια του Byte

Αν διαβάσουμε το αποτέλεσμα μαζί με το C8:
010010010 = +128+ 16+ 2 =146

Αν C7 διαφορετικό του C8 και ειδικότερα C7=1, C8=0 έχει συμβεί υπερχείλιση θετικού προσήμου. Ο αθροιστής (είναι μέρος της CPU) θα στείλει ένα σήμα το οποίο υποδηλώνει *OVERFLOW*.

Δηλαδή, το Δ=[-128..127]. Το αποτέλεσμά μας έχει υπερβεί τα όρια κατά 19 (146). Αυτό δεν φαίνεται. Η υπερχείλιση αφορά στο ΠΡΟΣΗΜΟ.

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Προσθέτουμε τα Α και Β στην κανονική τους μορφή. Άρα είναι 2 θετικοί αριθμοί, A7=B7=0
Ξαφνικά διαπιστώνουμε ότι το *S7=1*. Δηλαδή **προσθέσαμε 2 θετικούς και το άθροισμα είναι αρνητικό (ποτς γκένεν ατο;)**

S7=1.
Όμως S7=A7+B7+C7=0+0+C7.
Άρα για να συμβεί αυτό το σφάλμα πρέπει C7=1.
Αν C7=1 τότε η πράξη της αριστερότερης στήλης θα δώσει S7=1 και C8=0.
Άρα όταν C7=1, C8=0, υπάρχει σφάλμα και το πραγματικό πρόσημο του αποτελέσματος έχει υπερχειλίσει στη θέση 8, έξω από τα όρια του byte.

ΥΠΕΡΧΕΙΛΙΣΗ αφορά στο ΠΡΟΣΗΜΟ. Εδώ το θετικό πρόσημο έχει βγει έξω από τα όρια του ενός byte

0 10010010 (9 bit) = +146
(ΌΜΩΣ *δεν έχω 9 bit στη διάθεσή μου*. **Το πρόσημο δεν χωράει**)

Το αποτέλεσμα της πράξης είναι 146. Εμείς μπορούμε να χωρέσουμε ως και 127. Άρα η υπερχείλιση είναι 19; **ΌΧΙ είναι στο πρόσημο**!!!!

lect3PG5

-A-B=-96-50

96= 01100000
50= 00110010
-96= 10100000
-50= 11001110

	7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	ΘΕΣΕΙΣ
	1	0	1	0	0	0	0	0	Ai
	1	1	0	0	1	1	1	1	Bi
	0	0	0	0	0	0	0	0	Ci
1	0	1	1	0	1	1	1	0	Si

Προσθέτουμε τα συμπληρώματα των αριθμών 50, 96.

Το S7=0 δηλαδή προσθέσαμε 2 αριθμούς αρνητικούς και λάβαμε θετικό αποτέλεσμα! ΑΔΥΝΑΤΟ.

A7=B7=1. Άρα *για να είναι S7=0, θα πρέπει C7=0*
(**S7=A7+B7+C7 => S7=1+1+C7**).
Άρα, εκτελέστηκε η πράξη 1+1+0 και έδωσε S7=0 και C8=1.
Το C8=1 είναι το αρνητικό πρόσημο που έχει υπερχειλίσει.

Είναι C7=0, C8=1. Ο αθροιστής στέλνει μήνυμα σφάλματος.

Το αποτέλεσμα **1** 01101110 είναι ένας αρνητικός αριθμός
(αν λάβουμε υπόψη και το πρόσημο που υπερχείλισε). ΠΟΙΟΣ είναι;

Σ_2(101101110) = **0** 10010010 = 128+16+2= 146
Άρα το **1** 01101110= -146 με τη διαφορά ότι δεν χωράει σε 1 byte και το *πρόσημο έχει φύγει εκτός ορίων*.

Άρα όταν υπερχειλίσει αρνητικό πρόσημο,
ΠΑΝΤΑ ισχύει ότι C7=0, C8=1

-146= **1**1111111 01101110
(ΤΟ αποτέλεσμά μας ήταν μόνο το δεξιό byte)

Σ2 (**1**1111111 01101110) = 00000000 10010010 =+146

_____**ΤΜΗΜΑ ΠΕΜΠΤΗΣ**_____
A7=B7=1 αφού προσθέτουμε 2 αρνητικούς.

Για να βγει το αποτέλεσμα αρνητικό όπως αναμένεται πρέπει *C7=1*.
S7=A7+B7+C7= 1+1+C7 = 0+C7
Δηλαδή για να είναι **S7 = 1 πρέπει C7 = 1**.

Αν C7=0, τότε S7=A7+B7+C7=1+1+0 = 0 (ΘΕΤΙΚΟΣ ΑΡΙΘΜΟΣ) και C8=1.

Άρα, αν C7<>C8 και ειδικότερα C7=0, C8=1 έχουμε υπερχείλιση του αρνητικού προσήμου, το οποίο βρίσκεται στο C8

Αν το αποτέλεσμα το είχαμε λάβει με 2 byte 11111111 01101110, τότε αυτός είναι ο αριθμός -146. Όμως το -146 δεν χωράει σε 1 byte.

ΣΥΝΟΠΤΙΚΑ: Για κάθε πράξη μεταξύ των ακεραίων αριθμών Α,Β (εδώ σε 1 byte αλλά επεκτείνεται σε περισσότερα) ισχύει ΜΙΑ από τις εξής περιπτώσεις:

1. C7=C8=0, τότε το αποτέλεσμα είναι αρνητικό και το κρατούμενο C8 αγνοείται.
2. C7=1=C8, τότε το αποτέλεσμα είναι θετικό και το κρατούμενο C8 αγνοείται.

Σε αυτές τις 2 περιπτώσεις **ΔΕΝ έχουμε υπερχείλιση**

3. C7=0, C8=1, τότε το αποτέλεσμα είναι εσφαλμένο (Υπερχείλιση) και το αρνητικό πρόσημο βρίσκεται στη θέση C8. Το αποτέλεσμα που εμφανίζεται ως άθροισμα έχει S7=0 και είναι θετικό. ΠΡΟΣΤΕΘΗΚΑΝ ΑΡΝΗΤΙΚΟΙ ΑΡΙΘΜΟΙ ΚΑΙ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΘΕΤΙΚΟ

4. C7=1, C8=0, τότε το αποτέλεσμα είναι εσφαλμένο (Υπερχείλιση) και το θετικό πρόσημο βρίσκεται στη θέση C8. Το αποτέλεσμα που εμφανίζεται ως άθροισμα έχει S7=1 και είναι αρνητικό. ΠΡΟΣΤΕΘΗΚΑΝ ΘΕΤΙΚΟΙ ΑΡΙΘΜΟΙ ΚΑΙ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΑΡΝΗΤΙΚΟ

ΣΥΜΠΕΡΑΣΜΑΤΙΚΑ υπερχείλιση εντοπίζεται όταν C7<>C8

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Προσθέτω 2 αρνητικούς και λαμβάνω αποτέλεσμα θετικό.
Είναι S7=0
Όμως S7=A7+B7+C7=1+1+C7. Άρα για να συμβεί αυτό το σφάλμα πρέπει C7=0
Τότε, θα είναι **S7=0 και C8=1**.

Άρα όταν C7=0 και C8=1, υπάρχει σφάλμα υπερχείλισης προσήμου. Το αρνητικό πρόσημο έχει υπερχειλίσει στη θέση C8.

ΤΕΛΙΚΑ: Υπερχείλιση υπάρχει όταν C7<>C8

lect4PG1

ΑΡΙΘΜΟΙ ΚΙΝΗΤΗΣ ΥΠΟΔΙΑΣΤΟΛΗΣ

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

$37.75 = 100101.11 \times 2^0 = 100101.11 \times 1$

Έστω τώρα ότι θέλουμε να φέρουμε τον αριθμό στη μορφή 1.xxxxxx δηλαδή να μετακινήσουμε την υποδιαστολή τόσες θέσεις έτσι ώστε να έχουμε τον αριθμό στη μορφή 1.xxxxx

Εδώ, η υποδιαστολή πρέπει να μετακινηθεί κατά E = 5 θέσεις αριστερά. Αυτό θα έχει ως συνέπεια ο εκθέτης να αυξηθεί κατά 5, δηλαδή από 0 να γίνει 5.

$37.75 = 1.0010111 \times 2^5$

$(1+ 1/8 + 1/32 + 1/64 + 1/128) \times 32 = 37.75$

$1100.0 = 12.0 \times 2^0$

$1.100 \times 2^3 = 1.5 \times 8 = 12$

$12 \times 1 = 1.5 \times 8$

Στο δεκαδικό, $120.0 = 120 \times 1 = 1.2 \times 100$ (1.2×10^2)

Ερώτηση: Γιατί να φέρουμε τον αριθμό στη μορφή 1.xxxxxx

ΑΠΑΝΤΗΣΗ: ΟΛΟΙ οι αριθμοί κινητής υποδιαστολής αποθηκεύονται στην ΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΗ μορφή 1.xxxx γιατί με τον τρόπο αυτό το Μπορούμε να μην το αποθηκεύσουμε (αλλά να το αφήσουμε να υπονοείται) σώζοντας ένα bit. Ένα bit επιπλέον παρέχει διπλάσια ακρίβεια.

$0.75 = 0.11 \times 2^0$.

Για να έρθει στη μορφή 1.xxx πρέπει η υποδιαστολή να μετακινηθεί ΜΙΑ ΘΕΣΗ ΔΕΞΙΑ (ο εκθέτης μειώνεται κατά μία θέση),
Άρα $1.1 \times 2^{-1} = 1.5 / 2 = 0.75$

Εδώ ο εκθέτης είναι αρνητικός.

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗ_____

$20.0 = 10100.0 \times 2^0$
(ο εκθέτης είναι ίσος με 0, και $2^0 = 1$)

Έστω ότι θέλουμε να αναπαραστήσουμε τον ίδιο αριθμό στη μορφή 1.xxxx

Τότε, ο αριθμός αυτός θα γραφτεί ως εξής: 1.01000 (η υποδιαστολή μετακινείται 4 θέσεις αριστερά). Αυτό έχει ως συνέπεια ο εκθέτης από 0 να γίνει 4 δηλαδή να αυξηθεί κατά 4

$20.0 = 1.01000 \times 2^4 = 1.25 \times 16$

Ο εκθέτης αυξήθηκε κατά Κ όπου Κ είναι το πλήθος των θέσεων κατά τις οποίες μετακινήθηκε αριστερά η υποδιστολή.

33.75= 100001.11 x 20

Αν ο αριθμός αυτός γραφτεί στη μορφή 1.xxxx θα γίνει

1.000011 και η υποδιαστολή μετακινείται 5 θέσεις αριστερά, άρα ο εκθέτης είναι 5

$33.75 = 1.000011 \times 2^5 = 1 + 1x2^{-5} + 1x2\text{-}6 = (1 + 1/32 + 1/64) \times 32 = 33.75$

$0.625 = 0.101 \times 2^0$

Για να γραφτεί στη μορφή 1.xxxx πρέπει η υποδιαστολή να μετακινηθεί μία θέση ΔΕΞΙΑ.
Θα γίνει 1.01×2^{-1}

$1.01 \times ½ = 1.25 / 2 = 0.625$

Αριστερή μετατόπιση υποδιαστολής κατά Κ θέσεις: Εκθέτης αυξάνεται κατά Κ
Δεξιά μετατόπιση υποδιαστολής κατά Κ θέσεις: Εκθέτης μειώνεται κατά Κ

ΓΙΑΤΙ στη μορφή 1.xxxxxx

Γιατί αν όλοι οι πραγματικοί αριθμοί αποθηκευτούν στη μορφή 1.xxxxxx τότε το κομμάτι 1. μπορεί να ΜΗΝ αποθηκευτεί αλλά να υπονοείται.
Αυτό σημαίνει ότι γλιτώνουμε 1 bit.

ΟΛΟΙ οι αριθμοί κινητής υποδιαστολής αποθηκεύονται στη μορφή 1.xxxxxxx αλλά το 1. ΔΕΝ ΑΠΟΘΗΚΕΥΕΤΑΙ.
Εκθέτης δείχνει το πόσο μετακινείται η υποδιαστολή.

lect4PG2

ΠΡΟΤΥΠΟ IEEE 754

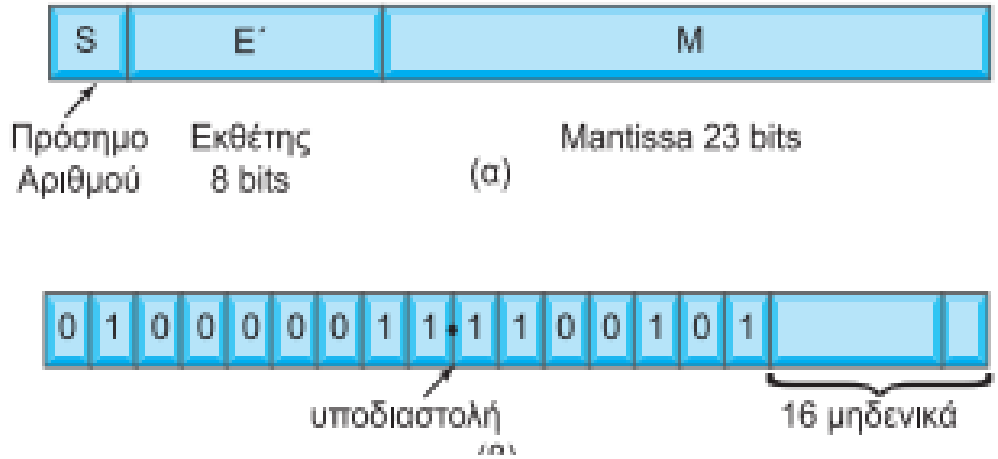
2 τμήματα: Εκθέτης, ορίζει το τμήμα της υποδιαστολής

Κλασματικό τμήμα (mantissa)

Απλή ακρίβεια (32 bits)

Διπλή Ακρίβεια (64 bits)

lect4PG3



Αριθμός 28.625
ΣΗΜΑΝΤΙΚΟ: Αν ο εκθέτης είναι 4, γράφεται το Ε'=127+4 (Υπέρβαση 127)
Γενική μορφή **1.M x 2 $^{E'-127}$**

S = Πρόσημο, S=1 για αρνητικούς αριθμούς και 0 για θετικούς.
Ε'= έχει μέγεθος 8 bit και είναι μη προσημασμένος (άρα θετικό).
½ = 2^{-1}
Πως αναπαρίστανται αρνητικοί.
Βρίσκουμε την τιμή του εκθέτη, Ε, και εφαρμόζουμε υπέρβαση 127, δηλαδή αντί του Ε γράφουμε Ε'=Ε+127
ΓΙΑΤΙ;;;;;;;

Π.χ. αν ο εκθέτης Ε βρεθεί (ΘΑ ΔΟΥΜΕ ΜΕ ΠΟΙΟΝ ΤΡΟΠΟ) ίσος με -2, στο πεδίο Ε θα γραφτεί 125
Αν βρεθεί ότι Ε= -5, θα γραφτεί 122
Αν Ε=3, θα γραφτεί 130

Θέλουμε τον αριθμό 28.625. σε αναπαράσταση απλής ακρίβειας.

1) Γράφουμε τον αριθμό σε δυαδική μορφή (απλή μετατροπή):
11100.101
12.0= 1100.0 = 1100.0 x 2^0 = 1100.0 x 1

Αν μετακινήσω την υποδιαστολή, έτσι ώστε **ο αριθμός μου να έρθει στη μορφή 1.xxx, τότε στο συγκεκριμένο παράδειγμα πρέπει η υποδιαστολή να μετακινηθεί Ε=3 θέσεις αριστερά**. Αυτό σημαίνει ότι ο εκθέτης μου θα είναι ίσος με 3.

12.0 = 1.1000 x 2^3 = 1.5 x 8 =12

18.75 = 10010.11 = 10010.11 x 2^0 = 10010.11 x 1.
Για να το φέρω στη μορφή 1.xxxxx πρέπει να μετακινήσω την υποδιαστολή κατά 4 θέσεις, δηλαδή Ε=4

18.75 = 1.001011 x 2^4 = 1.171875 x 16 = 18.75

0.5 = 0.1 Για να φέρουμε την ποσότητα αυτή στην μορφή 1.xxx πρέπει **να μετακινήσουμε την υποδιαστολή ΔΕΞΙΑ κατά Ε=1 θέσεις, άρα Ε=-1**

$1.0 \times 2^{-1} = 1. ½ = 0.5$

ΓΕΝΙΚΗ ΜΟΡΦΗ των αριθμών κινητής υποδιαστολής είναι $1.xxxxx \times 2^{E}$

ΓΙΑΤΙ 1.xxxxx;
Γιατί το 1. ΔΕΝ αποθηκεύεται αλλά υπονοείται και επομένως σώζω ένα bit διπλασιάζοντας το εύρος αριθμών που μπορώ να πετύχω

2) Φέρνω τον αριθμό σε μορφή 1.xxxx και ορίζω την τιμή του εκθέτη.
3) Γράφω το Ε'=Ε+127
4) Αποθηκεύω το κλασματικό μέρος ΧΩΡΙΣ το 1.xxxxx

28.625= 11100.101

2) Φέρνω τον αριθμό στη μορφή 1.xxxx και ορίζω τον εκθέτη. Εδώ 1.1100101 και Ε=4
3) Άρα Ε'=131
4) 1100101 0000000000000000

S	E'	M
0	10000011	1100101 (και 16 μηδενικά)

Δίνεται η αναπαράσταση της διαφάνειας. Να βρείτε τον αριθμό

- Είναι **θετικός** S=0
- Ο **εκθέτης είναι 10000011 = 131**, άρα ο πραγματικός εκθέτης είναι **131-127=4**
- Η mantissa είναι **1100101** και τοποθετούμε μπροστά το **1.** το οποίο ΔΕΝ ΑΠΟΘΗΚΕΥΕΤΑΙ

Δηλαδή **1.1100101×2^{4}**

Άρα **αφού Ε=4 μετακινώ την υποδιαστολή 4 θέσεις δεξιά:**
11100.101 = 28.625

ΔΕΥΤΕΡΟ ΠΑΡΑΔΕΙΓΜΑ 129.75

1) 129.75 = 10000001.11
2) Για να έρθει στη μορφή 1.xxxxx πρέπει να μετακινήσω την υποδιαστολή κατά Ε=7 θέσεις άρα Ε=7, 1.000000111
3) Ε'=127+7=134

S	E'	M
0	10000110	00000011100000000000000

Να αναπαραστήσετε το 1 και το 0

1= 1.0 x 2^0
E=0, επομένως E’=127

S	E’	M
0	01111111	23 μηδενικά

Το 0 συμβαίνει να έχει την ίδια αναπαράσταση!!!!!!
Για να αποθηκευτεί το 0, E’=00000000

S	E’	M
0	00000000	23 μηδενικά

_____ΤΜΗΜΑ ΠΕΜΠΤΗΣ_____

S = 0 ή 1

E’ : Ανεξάρτητα από την τιμή του εκθέτη E, αυτό που αποθηκεύεται είναι η λεγόμενη ΥΠΕΡΒΑΣΗ-127. E’=E+127.
Θέλουμε μέσα στο πεδίο εκθέτη να **αποθηκεύονται ΑΠΟΚΛΕΙΣΤΙΚΑ θετικοί αριθμοί** (Ο λόγος θα παρουσιαστεί στη συνέχεια και έχει να κάνει με τη σύγκριση αριθμών).

Π.χ Αν υπολογίσουμε ότι E=5, θα γραφτεί E’=132
Αν υπολογίσουμε ότι E=-1, θα γραφτεί E’=126

Υπέρβαση-127 = Excess-127
(Η πραγματική τιμή του εκθέτη αυξάνεται κατά 127)

Άρα στο E’ γράφουμε μη προσημασμένους θετικούς.

Αν E’=255, E=128 (ΑΚΡΑΙΑ)

Χρειάζεται να ξέρουμε:
1) Στη μάντισσα δεν αποθηκεύεται το 1. αλλά υπονοείται και
2) ότι στον εκθέτη E’ εφαρμόζεται υπέρβαση.

28.625, να γραφτεί σε Excess-127

Πρόσημο S=0.
Μετατρέπουμε το 28.625 σε δυαδικό (μη προσημασμένο, όχι σε επίπεδο byte, όσα bit χρειάζονται)
28.625 = 11100.101
3) **Φέρνουμε τον αριθμό στη μορφή 1.xxxx υπολογίζοντας το E**
1.1100101 και το E = 4 (μετακίνηση υποδιαστολής 4 θέσεις αριστερά)
4) Υ**πέρβαση E’=131.**
5) **Τελική αναπαράσταση** (το 1. που εμφανίζεται με έντονα ΔΕΝ αποθηκεύεται)

S	E′	M
0	10000011	11001010000000000000000

Δίνεται η αναπαράσταση της διαφάνειας. Να βρούμε τον αριθμό

S=0 άρα θετικός
E′=10000011 = 131 ότι το πραγματικό E (οι θέσεις μετακίνησης της υποδιαστολής είναι **131-127**). Επειδή **4>0 η μετακίνηση έγινε αριστερά**.
Η mantissa είναι **1100101**
4) **Προσθέτω το 1. στη μάντισσα. Άρα 1.1100101**
5) **Επειδή η υποδιαστολή μου μετακινήθηκε 4 θέσεις αριστερά, για να βρω τον αρχικό αριθμό την πάω 4 θέσεις δεξιά**. Δηλαδή
11100.101 = 28.625

E′ δείχνει πόσες θέσεις μετακινήθηκε η υποδιαστολή δεξιά ή αριστερά προσαυξημένες κατά 127

__ Έστω ο αριθμός 100.5. Αναπαράσταση σε excess-127_____

$100.5 = 1100100.1 \times 2^0$

Μετακινώ την υποδιαστολή 6 θέσεις αριστερά για να φέρω τον αριθμό στη μορφή 1.xxxxx
Δηλαδή $100.5 = 1.1001001 \times 2^6$

E′=133

S	E′	M
0	10000101	10010010000000000000000

Αναπαράσταση του 1.

$1.0 = 1.0 \times 2^0$
Εκθέτης =0
Άρα E′=127

S	E	M
0	01111111	23 μηδενικά

Το πρόβλημα είναι ότι ο αριθμός **0 γράφεται 0.0** και έχει την ίδια αναπαράσταση.

S	E′	M
0	00000000	23 μηδενικά

Το **0 αναπαρίσταται με τιμή E=-127, άρα όταν γίνει η υπέρβαση θα γραφτεί ως 0**.

S = 0 ή 1
E’: Όταν βρεθεί η τιμή του εκθέτη, αντί αυτής της τιμής γράφεται μία υπέρβαση κατά 127. Αν ο εκθέτης είναι E, στον υπολογιστή αποθηκεύεται E’=E+127
(Excess-127).

Ο λόγος που γίνεται αυτό είναι επειδή θέλουμε όλοι οι εκθέτες να είναι **ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΙ θετικοί**.
Αν πχ E=4, γράφεται E’=131
Αν E= -20 (υποδιαστολή 20 θέσεις δεξιά) τότε E’=107.

Με ένα byte οι προσημασμένες τιμές θα ήταν από
[-128...127] (Εδώ βρίσκονται τα E).
Οι μη προσημασμένες θα ήταν από
[0…255] (ΕΔΩ βρίσκονται τα E’)
Δεν χρησιμοποιούμε υπέρβαση 128 γιατί μερικές τιμές του E’ δεσμεύονται για ειδικούς σκοπούς.

ΘΕΛΟΥΜΕ οι εκθέτες που αποθηκεύονται να είναι ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΙ

Πράγματα που πρέπει να θυμηθούμε:
1) Το 1. δεν αποθηκεύεται,
2) Ο εκθέτης γράφεται προσαυξημένος κατά 127.

Να δείξουμε την αναπαράσταση excess-127 του αριθμού 28.625

1) Μετατρέπουμε τον αριθμό στο δυαδικό (μη προσημασμένου, όχι σε επίπεδο byte, δηλαδή χρησιμοποιώ όσα Bit είναι απαραίτητο)

11100.101

2) Γράφουμε τον αριθμό στη μορφή 1.xxxx και υπολογίζουμε τα E, E’

1.<u>1100101</u>, άρα E=4 και E’=131

3) Γράφω την τελική αποθήκευση. Τα έντονα γράμματα δεν αποθηκεύονται

S	E’=131	M
0	10000011	11001010000000000000000

Δίνεται η αναπαράσταση της διαφάνειας.
Εσωτερικά η μνήμη αποθηκεύει τα δεδομένα του σχήματος (β). Ποιος είναι ο αριθμός.

S=0. Άρα ξέρουμε ότι είναι *θετικός* αριθμός
Ο εκθέτης E’ είναι **131 άρα E=131-127 =4** άρα υπάρχει μετακίνηση της υποδιαστολής *4 θέσεις αριστερά*.
Κατασκευάζουμε τη μάντισσα προσθέτοντας το 1. που δεν αποθηκεύεται. Δηλαδή η μάντισσα είναι **1.1100101**
και ο αριθμός μου είναι ο **1.1100101×2^4**

4) **Για να βρω τον αριθμό, πρέπει να μετακινήσω την υποδιαστολή 4 θέσεις δεξιά** άρα θα είναι 11100.101= 28.625

Να αναπαρασταθεί το 0 και το 1

1: 1.0 x 20 άρα είναι στη μορφή που θέλουμε. E=0 άρα E’=127

S	E	M
0	01111111	23 μηδενικά

0: 0.0 x 20, ίδιο με το 1, E=0 άρα E’=127

Συμβατικά, **το 0 αποθηκεύεται με τιμή εκθέτη E=-127, άρα E’=0**

S	E’	M
0	00000000	23 μηδενικά

ΣΥΜΠΕΡΑΣΜΑ: Η τιμή E’=0 δηλαδή E=-127 χρησιμοποιείται ΜΟΝΟ για την αποθήκευση του μηδέν.

lect4PG4

Τιμές εκθέτη

- E’=255 (E=128, για αυτό δεν έχουμε υπέρβαση 128 αλλά 127),
- Mantissa <>0 μη αριθμός (π.χ. αρνητική ρίζα). 255=11111111
- E’=255, Mantissa=0 (διαίρεση με 0)
- Αναπαράσταση του 1 θα δώσει τιμή εκθέτη 127 και mantissa 0. Το ίδιο και το 0. Τι κάνουμε σε αυτή την περίπτωση;
- E’=0, M=0

Οι τιμές εκθέτη E’, 0 και 255 είναι ειδικές.
Για κανονικούς αριθμούς το E’ είναι από 1 ως 254

E = -126 ως 127

Αν E=-126 θα γραφτεί E’=1
Αν E=127 τότε E’=254
M=mantissa

Μπορούμε να έχουμε θετικούς αριθμούς μικρότερους του 1, με ακρίβεια ως και $2^{-126}= 1/2^{126}$
Μπορούμε ΠΑΛΙ να έχουμε σφάλμα περικοπής, δεν καλύπτει το σύνολο των αριθμών
Μπορούμε να έχουμε θετικούς αριθμούς μέχρι και $1.M \times 2^{127}$
Μπορούμε να έχουμε αρνητικούς αριθμούς μέχρι και $-1.M \times 2^{127}$

$+1.M \times 2^{E}$ E από [-126…. 127]
$-1.M \times 2^{E}$ E από [-126…. 127]

Μπορούμε να έχουμε ΤΕΡΑΣΤΙΑ ΑΚΡΙΒΕΙΑ αλλά ΠΟΤΕ δεν θα μπορέσουμε να αποθηκεύσουμε με ακρίβεια όλους τους αριθμούς. Πάντα θα υπάρχουν αριθμοί με μικρά σφάλματα περικοπής.

lect4PG6

ΣΥΓΚΡΙΣΕΙΣ ΑΡΙΘΜΩΝ

Έστω σύγκριση του 86 με το 54

A= 01010110
B= 00110110

Ξεκινάω από αριστερά, αν τα bit αριστερά είναι ίδια τότε προχωράω στο επόμενο (0=0)
Στο επόμενο έχω 1 και 0 άρα **ο αριθμός που έχει τη μονάδα είναι μεγαλύτερος**
Επειδή A7=B7=0 συγκρίνω τα A6, B6. Επειδή A6=1, B6=0, A>B

Αν συγκρίνω 86 με -86:
επειδή το πρώτο bit του 86 είναι 0
ενώ του -86 είναι 1, 86>-86
A= 01010110
B= 10101010
Επειδή A7=0, B7=1, A>B

-86 με το 54 (ίδια περίπτωση με τη δεύτερη)

Αν είναι αρνητικοί αριθμοί,

A= 10101010 = -86
B= 11001010 = -54

Αν έχω να συγκρίνω θετικούς ή αρνητικούς αριθμούς ή συνδυασμό των 2 πρέπει να χρησιμοποιήσω συγκριτική μεγέθους, ο οποίος είναι αρκετά πολύπλοκος διότι πρέπει να λάβει υπόψη πολλές διαφορετικές περιπτώσεις.
A7=B7, άρα προχωρώ στα A6, B6. Επειδή A6=0 B6=1, το B>A

Αν είχα τρόπο να συγκρίνω ΑΠΟΚΛΕΙΣΤΙΚΑ θετικούς αριθμούς, η σύγκριση θα ήταν ταχύτερη γιατί θα είχα μόνο μία περίπτωση.

Τελικά, η ταχύτερη σύγκριση γίνεται όταν οι αριθμοί είναι ΜΗ ΠΡΟΣΗΜΑΣΜΕΝΟΙ θετικοί.

ΓΙΑ ΝΑ ΓΙΝΕΙ ΠΡΟΣΘΕΣΗ ή ΑΦΑΙΡΕΣΗ ΑΡΙΘΜΩΝ ΚΙΝΗΤΗΣ ΥΠΟΔΙΑΣΤΟΛΗΣ απαιτείται σύγκριση των εκθετών και εξίσωση. Για αυτό χρησιμοποιείται η υπέρβαση για να είναι οι εκθέτες θετικοί.

Έστω ότι A=86, B=54

A= 01010110
B= 00110110

Συγκρίνω Α7 με Β7 είναι ίσα;
Πάω στο Α6 με το Β6.
Επειδή Α6=1, Β6=0, A>B

Το ίδιο αν οι αριθμοί είναι μη προσημασμένοι, δηλαδή θετικοί.

A= 86 = 01010110
B= -54 = 11001010

Θα έπρεπε το υλικό μου **να ελέγξει τα 2 αριστερά bit** και αν είναι διαφορετικά να πει πχ A>B

Αν οι αριθμοί ήταν αρνητικοί

A= -86 = 10101010
B= -54 = 11001010

Α7=Β7, προχωράω και επειδή Β6=1 ενώ Α6=0, είναι B>A

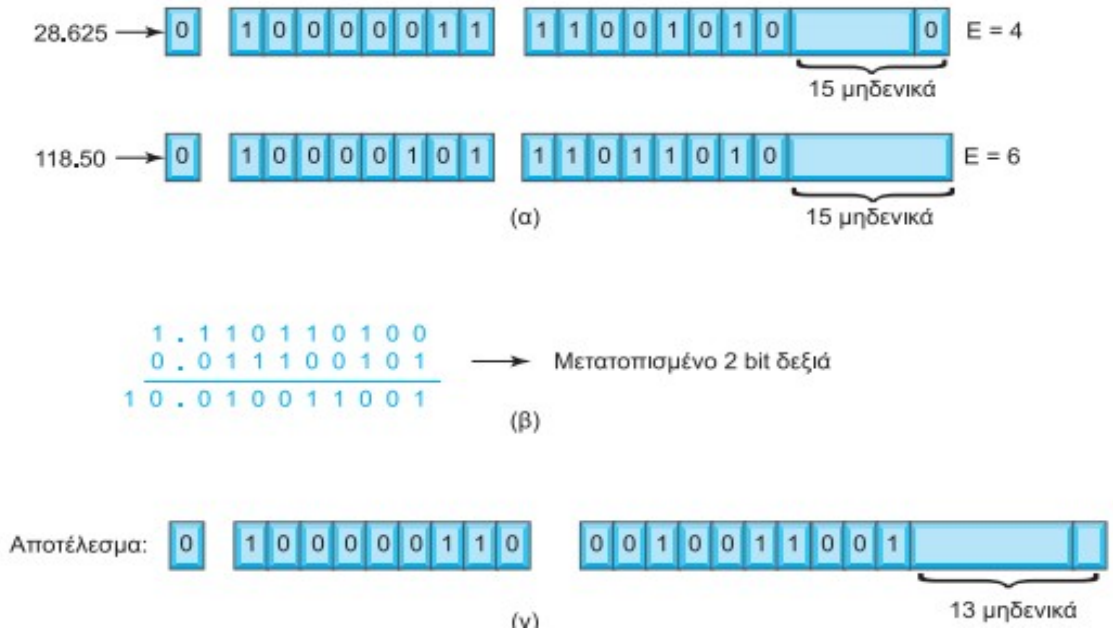
Είναι σαφώς ταχύτερο να συγκρίνουμε αποκλειστικά θετικούς μη προσημασμένους αριθμούς.

ΕΠΕΙΔΗ οι προσθέσεις και αφαιρέσεις κινητής υποδιαστολής απαιτούν οι εκθέτες να συγκρίνονται και να εξισώνονται, χρησιμοποιείται υπέρβαση και γράφονται αποκλειστικά σε θετική μορφή.

ΠΡΟΣΘΕΣΗ
Πρέπει να συγκριθούν οι εκθέτες και το κλασματικό μέρος με τον μικρότερο εκθέτη να μετακινηθεί δεξιά, όσες θέσεις είναι η διαφορά των εκθετών

lect4PG8

ΠΡΟΣΘΕΣΗ ΜΕ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ



28.625=

S	E	M
0	10000011	11001010

118.50 : 1110110.1, άρα για να έρθει στη μορφή 1.xxxx είναι E=6

Για να γίνει πρόσθεση:
ΛΑΜΒΑΝΕΙ ΜΕΡΟΣ ΚΑΙ ΤΟ 1.

Συγκρίνουμε τους εκθέτες των 2 αριθμών και φέρνουμε τον αριθμό με τον μικρότερο εκθέτη στην κατάλληλη μορφή, ώστε οι εκθέτες να εξισωθούν

28.625 έχει *εκθέτη 4* και πρέπει ο εκθέτης να γίνει 6. Άρα, στην αναπαράσταση πρέπει να προστεθούν αριστερά δύο μηδενικά.

$1.11001010 \times 2^4 = 0.0111001010 \times 2^6$

Στη συνέχεια προσθέτουμε τα κλασματικά μέρη

1.1101101000+
0.0111001010

Αποτέλεσμα: 10.010011001 (ΔΕΝ ΕΊΝΑΙ ΣΕ ΚΑΝΟΝΙΚΗ ΜΟΡΦΗ, 1.xxxxx)
Ο άσσος είναι κρατούμενο,

Το αποτέλεσμα είναι 0.010011001.
Για να γίνει κανονικοποίηση πρέπει ο άσσος που εμφανίζεται ως

κρατούμενο της πρόσθεσης να γίνει μέρος του αποτελέσματος (η υποδιαστολή πρέπει να πάει άλλη μία θέση αριστερά). Άρα ο εκθέτης θα γίνει 7

Το αποτέλεσμα είναι 1.0010011001.

S	E'=134 (127+7)	M
0	10000110	00100110010000000000000

ΤΜΗΜΑ ΠΕΜΠΤΗ

28.625 = 11100.101, που σημαίνει ότι για να γραφτεί στη μορφή 1.xxxxx πρέπει *E=4*

118.5 = 1110110.1, που σημαίνει ότι για να γραφτεί στη μορφή 1.xxxxx πρέπει *E=6*

Η διαφορά των 2 εκθετών είναι *2* (6-4). Ο κανόνας της πρόσθεσης λέει ότι για να προστεθούν οι αριθμοί αυτοί πρέπει να γίνει σύγκριση των εκθετών και ο αριθμός με τον μικρότερο εκθέτη να γραφτεί με τέτοιον τρόπο ώστε ο εκθέτης του να εξισωθεί με εκείνον του άλλου αριθμού. Στο παράδειγμα, π αριθμός 28.625 πρέπει να γραφτεί έτσι ώστε ο εκθέτης του από 4 να γίνει 6.
Στην πρόσθεση λαμβάνει μέρος ΟΛΗ η μάντισσα δηλαδή ΚΑΙ το κομμάτι 1. που δεν αποθηκεύεται (ΆΛΛΟ ΑΠΟΘΗΚΕΥΣΗ ΆΛΛΟ ΠΡΑΞΗ).

28.625 = 1.11001010 (η μάντισσα μαζί με το 1.)
Άρα 28.625 = 1.11001010×2^4

Για να αυξήσω τον εκθέτη σε 6 πρέπει να βάλω 2 μηδενικά αριστερά (ο εκθέτης αυξάνεται κατά 2).

28.625 = 0.0111001010×2^6

Εφόσον εξισώσουμε τους εκθέτες (**ΠΑΝΤΑ τον μικρότερο σε μεγαλύτερο**), *προσθέτουμε τις μάντισσες*

Μάντισσα του 118.5 = 1.11011010
Μάντισσα του 28.625 = 0.0111001010

ΑΠΟΤΕΛΕΣΜΑ: 1 0.010011001

Επομένως, το αποτέλεσμα είναι 0.010011001 ΔΕΝ ΕΊΝΑΙ ΚΑΝΟΝΙΚΟΠΟΙΗΜΕΝΟ. Για να κανονικοποιηθεί πρέπει να βάλουμε το κρατούμενο που παρήχθη από την πρόσθεση μέσα στο αποτέλεσμα. Άρα, η υποδιαστολή θα πάει μία θέση αριστερά, που σημαίνει ότι *E=6+1=7*

ΠΑΝΤΑ ΑΝ το αποτέλεσμα δεν είναι κανονικοποιημένο, υπάρχει κρατούμενο αριστερά (ΔΕΙΤΕ ΤΟ σε σχέση με τα bit A7, B7, C7, δεδομένου ότι ο ένας προσθετέος είναι 1.x).

ΤΕΛΙΚΗ ΑΠΟΘΗΚΕΥΣΗ:

S	E′ (127+7)	M
0	10000110	00100110010000000000000

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Ο κανόνας **πρόσθεσης δύο αριθμών κινητής υποδιαστολής** λέει ότι πρέπει

- πρώτα να συγκρίνουμε τους εκθέτες των 2 προστιθέμενων αριθμών.
*** Έστω ότι αυτοί είναι Κ και Λ. Αν Κ=Λ οι εκθέτες είναι εξισωμένοι άρα δεν χρειάζεται κάποια ενέργεια.**
*** Αν Κ>Λ τότε βρίσκω το Μ=Κ-Λ και στη συνέχεια παίρνω τη μάντισσα του αριθμού με τον μικρότερο εκθέτη Λ, και προσθέτω σε αυτή στο αριστερό μέρος Μ μηδενικά**

$12 = 1100.0 = 1.1 \times 2^3$
$0.11 \times 2^4 = 0.75 \times 16 = 12$
0.011×2^5
0.0011×2^6

- Αφού εξισώσουμε τους εκθέτες, προσθέτουμε τα κλασματικά μέρη (ΣΤΗΝ πρόσθεση λαμβάνει μέρος και το 1.)

$28.625 = 11100.101$ άρα $E=4 = 1.1100101 \times 2^4$
$118.50 = 1110110.1$ άρα $E=6 = 1.1101101 \times 2^6$

Είναι **Μ=Κ-Λ= 6-4 =2**

Άρα παίρνουμε τη μάντισσα του μικρότερου αριθμού (MAZI με το 1.) και τη γράφουμε

1.1100101×2^4
Για να γίνει ο εκθέτης 6 πρέπει να βάλουμε αριστερά 2 μηδενικά, 1
0.011100101

ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ: Το αποτέλεσμα είναι 0.010011001 και 1 μονάδα κρατούμενο. Αυτό το αποτέλεσμα δεν είναι στη μορφή 1.xxxx άρα απαιτείται κανονικοποίηση. Αυτό σημαίνει ότι το κρατούμενο 1 που θα υπάρχει πάντα αριστερά (ΓΙΑΤΙ;;;;;;) θα πρέπει να μπει μέσα στο αποτέλεσμα.
ΓΙΑΤΙ: Ένας αριθμός πάντα είναι 1.xxxx Αν ο άλλος συμβεί λόγω εξίσωσης εκθετών να είναι 0.000 και το αριστερό bit βγει μηδέν τότε το κρατούμενο της στήλης είναι 1. Άρα η πράξη ήταν 1+0+1 που θα δώσει 0 και 1 κρατούμενο

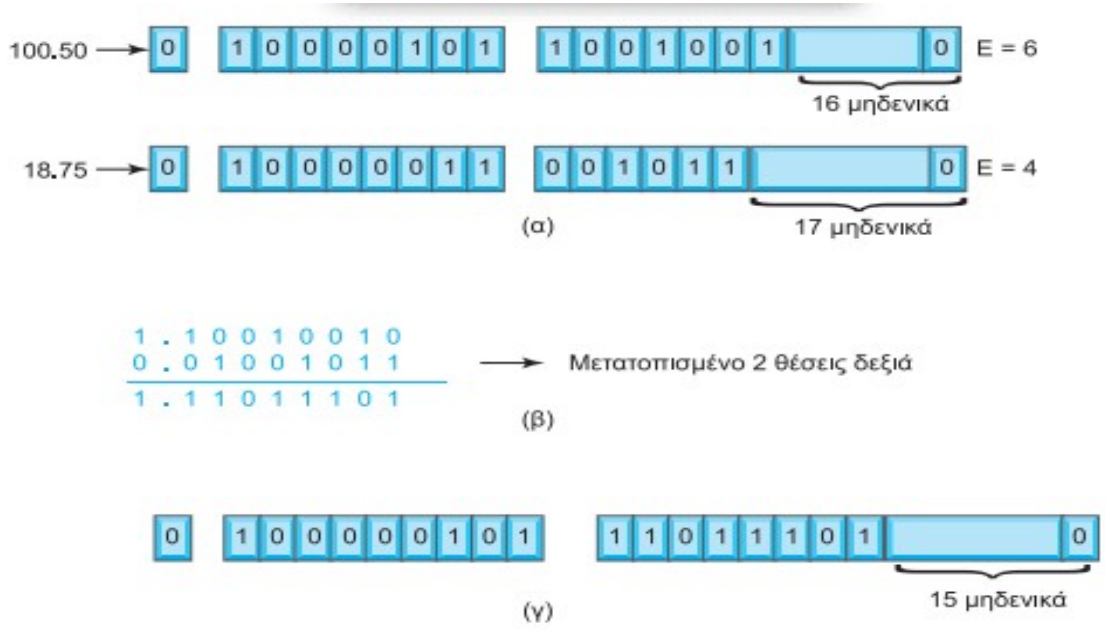
Αυτό σημαίνει ότι η υποδιαστολή θα πάει μία θέση αριστερά. Ο εκθέτης από 6 θα γίνει 7.

Άρα το αποτέλεσμα M= 1.0011100101 και E=7

S	E′ (127+7)	M
0	10000110	00100110010000000000000

lect4PG9

ΠΡΟΣΘΕΣΗ ΧΩΡΙΣ ΚΑΝΟΝΙΚΟΠΟΙΗΣΗ



$18.75 = 10010.11 \times 2^0$

$18.75 = 1.001011 \times 2^4$

$18.75 = 0.01001011 \times 2^6$

Ο αριθμός $18.75 = 1.001011 \times 2^4$
Έχει αποθηκευτεί με εκθέτη 4 (στο σχήμα α, Ε’=131, άρα Ε=4). Επειδή χρειάστηκε πρόσθεση με αριθμό που έχει εκθέτη 6, προστέθηκαν 2 μηδενικά αριστερά έγινε η πρόσθεση και στο τελικό αποτέλεσμα αποθηκεύεται εκθέτης 6. Στο σχήμα γ είναι Ε’=134, άρα Ε=6

η πρόσθεση 2 αριθμών με εκθέτη Κ και Λ όπου Κ>Λ θα δώσει αποτέλεσμα με εκθέτη Κ αν δεν χρειάζεται κανονικοποίηση, Κ+1 αν χρειάζεται. Ωστόσο, οι 2 προσθετέοι αποθηκεύονται με εκθέτες Κ και Λ (συν την υπέρβαση).
1ο παράδειγμα Κ=4 Λ=6 και το αποτέλεσμα είχε Κ=7 (χρειάστηκε κανονικοποίηση)
1ο παράδειγμα Κ=6 Λ=4 και το αποτέλεσμα είχε Κ=6

lect5PG1

ΛΟΓΙΚΕΣ ΠΥΛΕΣ

Λογική πύλη είναι το μικρότερο κομμάτι του υλικού. Είναι ένα μικρό κύκλωμα «προγραμματισμένο» να δίνει συγκεκριμένη τιμή εξόδου για κάθε συνδυασμό τιμών στην είσοδο.
Είσοδοι: Τα σήματα που τροφοδοτούν την πύλη και μπορεί να είναι **είτε ένα (για κάποιες πύλες) είτε 2 αλλά είτε και περισσότερα.**

Αν οι είσοδοι είναι Ν σε πλήθος δημιουργούν συνολικά 2Ν συνδυασμούς.
Αν έχω N=2 εισόδους αυτές σχηματίζουν 2^2=4 συνδυασμούς **(ΕΛΑΧΙΣΤΟΡΟΙ)**.
Αν N=3 έχουμε 8 συνδυασμούς, κ.ο.κ.
Αν N=1 έχουμε 2 συνδυασμούς.
Οι λογικές πύλες έχουν ΠΑΝΤΑ μία έξοδο. Όμως άλλα κυκλώματα μπορούν να έχουν περισσότερες εξόδους. Π.χ. ένα κύκλωμα αθροιστή έχει 2 εξόδους: Το άθροισμα S. Και το επόμενο κρατούμενο Cout
Επόμενο κρατούμενο είναι αυτό που παράγεται από την πρόσθεση ενός bit του αριθμού Α, ενός bit του αριθμού Β και ενός κρατούμενου C.

_____ΤΜΗΜΑ ΠΕΜΠΤΗ _____

Πως συμπεριφέρεται κάθε πύλη, δηλαδή ποια τιμή παράγει ως αποτέλεσμα ανάλογα με τις εισόδους που θα της δοθούν.
Πως συμβολίζεται κάθε πύλη.
Με ποια λογική συνάρτηση εκφράζουμε καθεμία από αυτές τις πύλες.

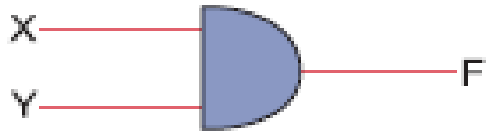
Πύλες= Μικρότερα κομμάτια του υλικού ενός Η/Υ. Εμφανίζουν μία προκαθορισμένη τιμή εξόδου (0 ή 1) ανάλογα με τον συνδυασμό των τιμών που δέχονται οι είσοδοί τους.

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

1) Αλγεβρική έκφραση
2) Τρόπο συμπεριφοράς
3) Επέκταση σε περισσότερες εισόδους (Στα σχήματα θα βλέπουμε 2 εισόδους αλλά θα δείξουμε και την επέκταση σε πολλές)
4) Σχηματική απεικόνιση

lect5PG2

ΚΑΙ = AND



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	0
1	0	0
1	1	1

Στο παράδειγμα βλέπουμε μία AND δύο εισόδων ΧΥ

Ν=2 είσοδοι και 2^2=4 συνδυασμούς εισόδων. Αν θέλω να γράψω αυτούς τους συνδυασμούς εύκολα, δεν έχω παρά να γράψω τους 4 αριθμούς από 0-3

X	Y	ΔΕΚΑΔΙΚΗ ΤΙΜΗ ΣΥΝΔΥΑΣΜΟΥ (Ελαχιστόρος)
0	0	0
0	1	1
1	0	2
1	1	3

Ο πίνακας αυτός, ο οποίος για κάθε συνδυασμό εισόδων δείχνει την τιμή της εξόδου λέγεται **ΠΙΝΑΚΑΣ ΑΛΗΘΕΙΑΣ**
ΟΙ πύλες ΚΆΘΕ στιγμή δέχονται ένα συνδυασμό εισόδων άρα δίνουν κάθε χρονική στιγμή μία συγκεκριμένη έξοδο. Αυτό μας δίνει τη δυνατότητα να μελετήσουμε τη συμπεριφορά τους με τη βοήθεια του πίνακα αληθείας.

ΠΥΛΗ ΚΑΙ

Η πύλη δίνει έξοδο 1 όταν όλες οι είσοδοι είναι 1.
Διαφορετικά, η ύπαρξη έστω ενός μηδενικού δίνει F=0

ΑΛΓΕΒΡΙΚΗ έκφραση: Κάθε έξοδος σε όλα τα κυκλώματα μπορεί να εκφραστεί αλγεβρικά.
ΠΩΣ;

Κοιτάζουμε σε ποιους συνδυασμούς εισόδων η **F=1**.
Για αυτούς τους συνδυασμούς εξετάζουμε τις εισόδους. **Αν μία είσοδος είναι 1 γράφεται σε κανονική μορφή αν είναι 0 σε συμπληρωματική.** *Οι συμπληρωματικές μορφές χρησιμοποιούν έναν τόνο (π.χ. Α')*

Γράφουμε κάθε συνδυασμό ως γινόμενο (ΛΟΓΙΚΟ ΚΑΙ).

Στο παράδειγμα της πύλης ΚΑΙ.
Ο συνδυασμός 3 είναι ο μοναδικός για τον οποίο F=1.
Για αυτό τον συνδυασμό οι είσοδοι Χ, Υ είναι 1, άρα θα γραφτούν στο γινόμενο με κανονική μορφή
Ο συνδυασμός είναι ΧΥ
Άρα **F= ΧΥ**

Τι δείχνει το ΧΥ (Αν Χ=1 ΚΑΙ Υ=1, τότε F=1)

ΣΥΜΠΛΗΡΩΜΑΤΙΚΟ: Αν κάποια χρονική στιγμή είναι Α=1, τότε την ίδια χρονική στιγμή το Α'=0 (ΥΠΑΡΧΕΙ ΠΥΛΗ ΑΝΤΙΣΤΡΟΦΕΑΣ)
Αν το Α=0, το Α'=1

ΥΠΟΘΕΣΗ
Έστω ο παρακάτω πίνακας αληθείας για 2 εισόδους

Χ	Υ	ΔΕΚΑΔΙΚΗ ΤΙΜΗ ΣΥΝΔΥΑΣΜΟΥ (Ελαχιστόρος)	Έξοδος F
0	0	0	0
0	**1**	1	**1**
1	0	2	0
1	**1**	3	**1**

F=;

Η F είναι 1 για 2 συνδυασμούς.
Οι 2 συνδυασμοί αυτοί ΔΕΝ μπορούν να ισχύουν μαζί.
Επομένως ή ο ένας ή ο άλλος.
Η F είναι 1 όταν (Χ= 0 ΚΑΙ Υ =1) Ή όταν(Χ=1 ΚΑΙ Υ=1)

F= Χ'Υ OR ΧΥ (*επειδή το OR Συμβολίζεται με* +), **F = Χ'Υ + ΧΥ**

Δηλαδή
Αν Χ=0 και Υ=1 τότε Χ'=1 και Υ=1 άρα Χ'Υ=1
Αν Χ=1 και Υ=1 τότε ΧΥ = 1.

ΠΟΤΕ Χ'Υ=1; 'Όταν Χ=0 ΚΑΙ Υ=1
ΠΟΤΕ ΧΥ=1; Όταν Χ=Υ=1

Έστω ένα κύκλωμα τριών εισόδων Α, Β, C και μίας εξόδου F.
- Να γράψετε όλους τους συνδυασμούς εισόδου
- Να δώσετε τη λογική έκφραση της F αν η F είναι 1 για τους συνδυασμούς 0, 2, 7

Παίρνω τις δεξιότερες εισόδους Β,C και γράφω τους 4 συνδυασμούς 00,01,10,11. Τους αντιγράφω από κάτω (διακεκομμένη). Για το πάνω τμήμα θέτω Α=0 για το κάτω Α=1.

A	B	C	Συνδυασμός	F
0	0	0	0	**1**
0	0	1	1	0
0	1	0	2	**1**
0	1	1	3	0
1	0	0	4	0
1	0	1	5	0
1	1	0	6	0
1	1	1	7	**1**

2) Η F Δίνει 1 όταν (**A=B=C =0**) Ή (**A=0, B=1, C=0**) Ή (**A=B=C=1**)

A=B=C=0-> A’ B’ C’
A=0, B=1, C=0 -> A’ B C’
A=B=C=1 -> ABC

F= A’ B’ C’ + A’ B C’ + ABC (*ΑΘΡΟΙΣΜΑ ΓΙΝΟΜΕΝΩΝ*). Αν ένα από τα γινόμενα είναι 1 (**ΜΟΝΟ 1 μπορεί αν είναι 1**), **F=1**

Έστω ότι τη χρονική στιγμή t, τα σήματα εισόδου ABC είναι 0, 1, 0 αντίστοιχα. Την χρονική στιγμή t ΔΕΝ ΜΠΟΡΕΙ ΝΑ ΕΧΟΥΝ ΆΛΛΕΣ ΤΙΜΕΣ παρά μόνο αυτές. Ποια είναι η έξοδος;

Αν *A=0, B=1, C=0* ΤΟΤΕ

A’ B’ C’ = 0’1’0’ = 1 0 1 = 0 (ΜΙΛΑΜΕ ΓΙΑ ΛΟΓΙΚΑ ΚΑΙ)
ABC = 010 =0 (0 AND 1 AND 0)
A’BC’ = 0’10’ =1

Άρα η **F = 0 + 1+ 0 =1** (*ΕΝΑΣ ΑΠΌ ΤΟΥΣ ΟΡΟΥΣ ΤΗΣ F είναι 1 κάθε φορά*)
Η F γίνεται 1 εξαιτίας ενός από τα γινόμενα κάθε φορά

Αν κάποια στιγμή A=0 B=0 C=1, τότε η F =0 γιατί κανένα από τα αλγεβρικά γινόμενα της F δεν δίνει 1.

Έστω ένα κύκλωμα με 4 εισόδους A,B C D και μία έξοδο F. Να γράψετε τους 16 συνδυασμούς.
Αν η **F=1 για τους συνδυασμούς 0, 7 και 15**, να γράψετε την αλγεβρική έκφραση της F

A	B	C	D	Συνδυασμός	F
0	0	0	0	0	1
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	0
0	1	0	1	5	0
0	1	1	0	6	0
0	1	1	1	7	1
1	0	0	0	8	0
1	0	0	1	9	0
1	0	1	0	10	0
1	0	1	1	11	0
1	1	0	0	12	0
1	1	0	1	13	0
1	1	1	0	14	0
1	1	1	1	15	1

F=A'B'C'D'+ A'BCD + ABCD

Έχουμε ένα άθροισμα τριών όρων γινομένου όπου κάθε όρος έχει 4 μεταβλητές

ΣΥΜΠΛΗΡΩΜΑΤΑ: Γράφονται για να εξασφαλίσω ότι ο αντίστοιχος συνδυασμός θα μου δώσει 1. Π.χ., από τον πίνακα αληθείας προκύπτει ότι μία από τις περιπτώσεις όπου η F=1 είναι όταν A=B=C=D=0. Άρα το αντίστοιχο γινόμενο θα είναι 1 όταν γραφτεί στη μορφή A’B’C’D’
ΠΥΛΗ ΚΑΙ ΠΟΛΛΩΝ ΕΙΣΟΔΩΝ: Δίνει **F=1 όταν ΟΛΕΣ οι είσοδοι είναι 1**.
Έστω μία AND με εισόδους X,Y,Z. Ποια η αλγεβρική έκφραση;
F= XYZ

X	Y	Z	F(Λογικό AND)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

F=XYZ

_____**ΤΜΗΜΑ ΠΕΜΠΤΗ** _____

Είναι μία πύλη η οποία δέχεται 2 ή και περισσότερες εισόδους. Η συμπεριφορά της είναι η εξής: **Αν έστω μία από τις εισόδους είναι 0, η έξοδος F είναι 0. Η F είναι 1 ΜΟΝΟ αν όλες οι είσοδοι είναι 1.**
Ο αριθμός εισόδων κάθε πύλης είναι προκαθορισμένος ανάλογα με την χρήση.

Πίνακας αληθείας. *Ο πίνακας που για κάθε συνδυασμό των εισόδων δείχνει την τιμή της εξόδου.*
Αν έχω N εισόδους σε ένα κύκλωμα, τότε έχω 2^N συνδυασμούς. Π.χ., αν N =2 έχουμε 4 συνδυασμούς. Αν N=3 έχουμε 8 συνδυασμούς, N=4 έχουμε 16 συνδυασμούς. *Για κάθε συνδυασμό, υπάρχει ΜΙΑ ΠΡΟΚΑΘΟΡΙΣΜΕΝΗ ΤΙΜΗ ΕΞΟΔΟΥ.*

Συνδυασμοί: ΠΩΣ τους γράφουμε; Αυτό ισχύει ΠΑΝΤΑ. Οι συνδυασμοί των εισόδων για όλα τα κυκλώματα παράγονται με τον ίδιο τρόπο.

ΠΑΡΑΔΕΙΓΜΑ: Να φτιάξετε τον πίνακα αληθείας των πυλών ΚΑΙ με 2, 3 και 4 εισόδους

2 είσοδοι: Έστω οι είσοδοι Α, Β. Πρέπει απλώς να σχηματίσουμε τους αριθμούς 0-3 (4 συνδυασμοί)

A	B	ΔΕΚΑΔΙΚΗ ΤΙΜΗ ΣΥΝΔΥΑΣΜΟΥ (Ελαχιστόρος)	Έξοδος F
0	0	0	0
0	**1**	1	0
1	0	2	0
1	**1**	3	**1**

2 είσοδοι σημαίνει 2 bit. Άρα γράψαμε με 2 bit τους αριθμούς 0-3. Οι συνδυασμοί εισόδων για κάθε κύκλωμα 2 Bit είναι οι ίδιοι, διαφέρει η συμπεριφορά δηλαδή η έξοδος.

3 είσοδοι: Πρέπει να έχουμε 8 συνδυασμούς, άρα γράφουμε τους αριθμούς 0-7. N=3 άρα 23 συνδυασμοί

A	B	C	Συνδυασμός	F
0	0	0	0	0
0	0	1	1	0
0	1	0	2	0
0	1	1	3	0
1	0	0	4	0
1	0	1	5	0
1	1	0	6	0
1	1	1	7	**1**

Ελαχιστόρος= Συνδυασμός τιμών των εισόδων.
Π.χ, αν A=1, B=0, C=1 αναφερόμαστε στον ελαχιστόρο 5, για τον οποίο F=0

Κάθε χρονική στιγμή, στις εισόδους εμφανίζεται ΜΟΝΟ ένας συνδυασμός. Άρα, για να μελετήσουμε τη συμπεριφορά πρέπει να βρούμε για καθένα από τους συνδυασμούς εισόδων πως θα συμπεριφερθεί η έξοδος, δηλαδή το κύκλωμα.
Π.χ. για την πύλη ΚΑΙ τριών εισόδων, οι είσοδοι μπορούν να βρεθούν κάθε χρονική στιγμή σε μία από τις 8 καταστάσεις (ελαχιστόροι). Αν κάποια χρονική στιγμή, οι είσοδοι βρεθούν στην κατάσταση 7 (ελαχιστόρος 7) όπου A=1 ΚΑΙ B=1 ΚΑΙ C=1 τότε F=1 διαφορετικά για

κάθε άλλη κατάσταση F=0

Αλγεβρικά, η AND τριών εισόδων ABC γράφεται F = ABC
Αλγεβρικά, η AND δύο εισόδων AB γράφεται F = AB

4 είσοδοι: A, B, C, D

A	B	C	D	Συνδυασμός	F
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	0
0	1	0	1	5	0
0	1	1	0	6	0
0	1	1	1	7	0
1	0	0	0	8	0
1	0	0	1	9	0
1	0	1	0	10	0
1	0	1	1	11	0
1	1	0	0	12	0
1	1	0	1	13	0
1	1	1	0	14	0
1	1	1	1	15	1

Για κάθε ελαχιστόρο πλην του 15, η F=0

_____**ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ**_____

Ν, είναι το πλήθος εισόδων ενός οποιουδήποτε κυκλώματος, αυτές οι είσοδοι σχηματίζουν 2^N διαφορετικούς συνδυασμούς (ελαχιστόροι).
Ουσιαστικά πρόκειται για 2^N διακριτές καταστάσεις στις οποίες μπορεί να βρεθεί το κύκλωμα. Για καθεμία από αυτές τις καταστάσεις υπάρχει μία τιμή της εξόδου.
Για όλα τα κυκλώματα, ο τρόπος καταγραφής των ελαχιστόρων είναι ίδιος.

ΠΥΛΗ ΚΑΙ 2 εισόδων: Έχουμε 2 εισόδους Χ, Υ άρα αυτές σχηματίζουν $2^2=4$ συνδυασμούς. 2 bit εισόδου. Αυτό που πρέπει να κάνουμε για τους 4 συνδυασμούς είναι να γράψουμε με 2 bit τους αριθμούς 0-3

A	B	ΔΕΚΑΔΙΚΗ ΤΙΜΗ ΣΥΝΔΥΑΣΜΟΥ (Ελαχιστόρος)	Έξοδος F
0	0	0	0
0	**1**	1	0
1	0	2	0
1	**1**	3	**1**

Η πύλη ΚΑΙ δίνει έξοδο 1 μόνο όταν όλες οι Ν είσοδοι είναι 1. Άρα μόνο για έναν συνδυασμό δίνει έξοδο 1. **Η ύπαρξη ενός μηδενικού στις εισόδους κάνει την F=0**
Πίνακας αληθείας: Δείχνει για κάθε συνδυασμό εισόδων (ελαχιστόρο) ποια είναι η τιμή της εξόδου.

Με 3 εισόδους, XYZ

A	B	C	Ελαχιστόρος	F
0	0	0	0	0
0	0	1	1	0
0	1	0	2	0
0	1	1	3	0
1	0	0	4	0
1	0	1	5	0
1	1	0	6	0
1	1	1	7	**1**

Έχουμε έξοδο 0 εκτός από την περίπτωση
όπου όλες οι είσοδοι είναι 1

4 είσοδοι

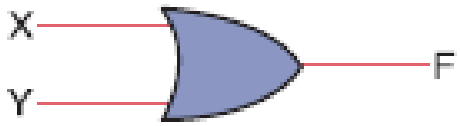
X	Y	Z	W	Συνδυασμός	F
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	2	0
0	0	1	1	3	0
0	1	0	0	4	0
0	1	0	1	5	0
0	1	1	0	6	0
0	1	1	1	7	0
1	0	0	0	8	0
1	0	0	1	9	0
1	0	1	0	10	0
1	0	1	1	11	0
1	1	0	0	12	0
1	1	0	1	13	0
1	1	1	0	14	0
1	1	1	1	15	1

Η AND δίνει 1 μόνο για τον συνδυασμό 1111, δηλαδή για τον ελαχιστόρο 15

F= XY
F= XYZ
F= XYZW

lect5PG3

ΠΥΛΗ OR = ή



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	1

OR δίνει 1 όταν μία έστω είσοδος είναι ίση με 1.
Αλγεβρικά γράφεται F=X+Y

F= X'Y +XY' + XY

Είναι ισοδύναμες οι εκφράσεις X'Y +XY' + XY(*προκύπτει από τον πίνακα*) και η **X+Y**

ΤΑΥΤΟΤΗΤΑ: X+X (X OR X) = X

F= X'Y + XY' + XY = (X'Y + XY) + (XY' + XY) (Αφού XY + XY = XY)
(1) (2)

(1)-> Y (X'+X) = Y AND 1 = Y
(2)-> X (Y'+Y) = X AND 1 = X

Άρα δικαιούμαι να πω ότι X'Y + XY' + XY = X+Y
Επέκταση σε πολλές εισόδους: Αν έστω μία είσοδος είναι 1, τότε F=1.

ΜΕ 4 εισόδους, έχω 16 συνδυασμούς. Από αυτούς μόνο ένας είναι ο 0000, όλοι οι άλλοι έχουν μία μονάδα. **Άρα 15 F= x+y+z+w**

_____ΤΜΗΜΑ ΠΕΜΠΤΗ_____
Αν έστω και μία από τις εισόδους είναι Αληθής (δηλαδή 1), η F = 1
Αλγεβρικά:
2 είσοδοι: F = X+Y = X OR Y
3 είσοδοι: F = X+Y+Z = X OR Y OR Z

Αν κάποια χρονική στιγμή οι είσοδοι X, Y σχηματίζουν τον ελαχιστόρο 1 Ή 2 Ή 3, τότε **F=1**
Προφανώς, η OR δίνει 0 μόνο όταν όλες οι είσοδοι είναι 0, δηλαδή σχηματίζουν τον ελαχιστόρο 0.

Π.χ. **OR 3 εισόδων**

A	B	C	Ελαχιστόρος	F
0	0	0	0	0
0	0	1	1	1
0	1	0	2	1
0	1	1	3	1
1	0	0	4	1
1	0	1	5	1
1	1	0	6	1
1	1	1	7	1

Διαφορά AND με OR: Οι συνδυασμοί εισόδων γράφονται παρομοίως ΓΙΑ ΟΛΕΣ ΤΙΣ ΠΥΛΕΣ, **η AND δίνει έξοδο 1 όταν όλες οι είσοδοι είναι 1**. Δηλαδή αν έχω Ν εισόδους πρέπει όλες να είναι 1 για να δώσει η F τιμή 1. Από την άλλη, **η OR με Ν εισόδους δίνει έξοδο 1 αν έστω μία από τις εισόδους είναι 1**.

_____ΤΜΗΜΑ ΠΑΡΑΣΚΕΥΗΣ_____

Δίνει έξοδο ένα, όταν μία έστω από τις εισόδους είναι ίση με 1 και μηδέν μόνο όταν όλες οι είσοδοι είναι 0. *Η έξοδος είναι 0 μόνο για τον ελαχιστόρο 0* ανεξαρτήτων του πλήθους των εισόδων.

Αλγεβρικά γράφουμε F=X+Y = X OR Y. Παρότι χρησιμοποιούμε το σύμβολο της πρόσθεσης και παρότι η πράξη AND που είδαμε είναι ισοδύναμη με τον πολλαπλασιασμό 2 bit, αυτό δεν είναι σωστό για την πράξη OR.

0 OR 0 = 0+0 = 0
0 OR 1 = 0+1 = 1
1 OR 0 = 1+0 = 1
1 OR 1 <> 1+1

Επειδή 1+1=10

Πίνακας αληθείας OR τριών εισόδων

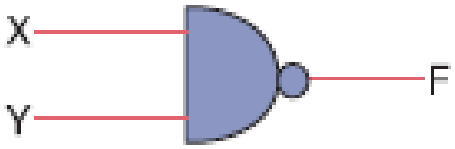
A	B	C	Ελαχιστόρος	F
0	0	0	0	0
0	0	1	1	1
0	1	0	2	1
0	1	1	3	1
1	0	0	4	1
1	0	1	5	1
1	1	0	6	1
1	1	1	7	1

Η F δίνει 0 μόνο για τον ελαχιστόρο 0. Για όλους τους άλλους 1.

OR τριών εισόδων (X+Y+Z)

lect5PG4

Πύλη ΟΧΙ-ΚΑΙ (NAND)



Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	1
1	0	1
1	1	0

Ο Συμβολισμός είναι ότι η AND Αλλά με άρνηση η οποία δηλώνεται με το κυκλάκι
Η NAND έχει αντίστροφη συμπεριφορά από την AND.
Δίνει F=1 όταν υπάρχει έστω ένα μηδενικό στις εισόδους, αλλιώς δίνει 0. ΑΝΤΙΣΤΡΟΦΗ ΤΗΣ AND

ΑΛΓΕΒΡΙΚΑ: F= (XY)’ Είναι η NAND

F = XY είναι η AND

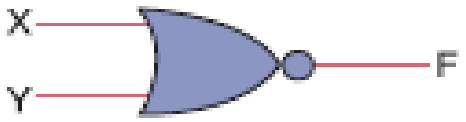
Είσοδοι		Έξοδος	Έξοδος
X	**Y**	**f=XY**	**F=(XY)’**
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Είσοδοι			AND Έξοδος	NAND Έξοδος
X	**Y**	**Z**	**F=XYZ**	**F=(XYZ)’**
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

Η NAND δίνει 1 για όλους τους ελαχιστόρους πλην του 7

lect5PG5

ΠΥΛΗ ΟΥΤΕ (NOR)



Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	0

NOR: Δίνει 1 μόνο όταν όλες οι είσοδοι είναι 0. ΑΝΤΙΣΤΡΟΦΗ της OR

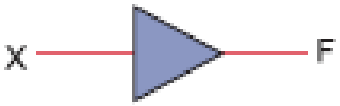
OR: F = X+Y
NOR: F = (X+Y)'
(X)' : Συμπλήρωμα του (X)
ΠΑΡΑΔΕΙΓΜΑ ΣΥΜΠΛΗΡΩΜΑΤΟΣ (X+Y+Z)' : Συμπλήρωμα του (X+Y+Z), δηλαδή αν ***X+Y+Z= 1 το (X+Y+Z)' = 0***

X	Y	F = X+Y	F=(X+Y)'
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Η NOR με περισσότερες εισόδους δίνει F=1 για τον ελαχιστόρο 0, για όλους τους άλλους 0.

lect5PG6

ΑΠΟΜΟΝΩΤΗΣ

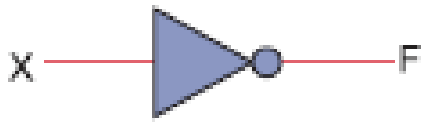


Είσοδος	Έξοδος
X	F
0	0
1	1

Έχει ΜΙΑ ΕΙΣΟΔΟ: Δίνει ως έξοδο ότι είναι η είσοδος F=X

lect5PG7

ΑΝΤΙΣΤΡΟΦΕΑΣ



Είσοδος	Έξοδος
X	F
0	1
1	0

Έχει ΜΙΑ ΕΙΣΟΔΟ: Αντιστρέφει ένα σήμα. F=X'

lect5PG8

ΠΩΣ ΒΡΙΣΚΟΥΜΕ ΤΗ ΛΟΓΙΚΗ ΕΚΦΡΑΣΗ ΑΠΌ ΤΟΝ ΠΙΝΑΚΑ ΑΛΗΘΕΙΑΣ

Έστω ένα κύκλωμα με 3 μεταβλητές εισόδου A, B, C, το οποίο δίνει έξοδο 1, όταν στις εισόδους εφαρμοστεί ένας από τους συνδυασμούς (ελαχιστόρους) 0, 1, 7. Διαφορετικά η έξοδος είναι 0. Να βρείτε την λογική (ή αλγεβρική) έκφραση που περιγράφει αυτό το κύκλωμα.

Καταρχήν κατασκευάζουμε τον πίνακα αληθείας

A	B	C	Ελαχιστόρος	F	
0	0	0	0	1	επειδή A=B=C=0 τα ABC γράφτηκαν συμπληρωματικά στο γινόμενο A'B'C'
0	0	1	1	1	επειδή A=B=0, C=1, AB συμπληρωματικά, C κανονικά A'B'C
0	1	0	2	0	
0	1	1	3	0	
1	0	0	4	0	
1	0	1	5	0	
1	1	0	6	0	
1	1	1	7	1	Επειδή όλα είναι 1, γράφτηκαν κανονικά ABC

ΛΕΚΤΙΚΑ: Το κύκλωμα δίνει F=1 αν οι είσοδοι είναι
(A,B,C) = (0,0,0) Ή
(A,B,C) = (0,0,1) Ή
(A,B,C) = (1,1,1)

(**A=0 ΚΑΙ B=0 ΚΑΙ C=0**) Ή
(**A=0 ΚΑΙ B=0 ΚΑΙ C=1**) Ή
(**A=1 ΚΑΙ B=1 ΚΑΙ C=1**)

Για να βγάλουμε τη λογική έκφραση της F ακολουθούμε τα εξής βήματα:
- Βρίσκουμε τους ελαχιστόρους για τους οποίους η F=1 και τους αναλύουμε σε γινόμενα (AND).

Ελαχιστόρος 0. Σημαίνει ότι **A=0 ΚΑΙ B=0 ΚΑΙ C=0**. Μπορώ αυτό να το γράψω ABC και αυτό το γινόμενο να είναι 1 όταν A=B=C=0
Ελαχιστόρος 1. Σημαίνει ότι **A=0 ΚΑΙ B=0 ΚΑΙ C=1**.
Ελαχιστόρος 7. Σημαίνει ότι **A=1 ΚΑΙ B=1 ΚΑΙ C=1**.

Η F είναι 1 όταν συμβαίνει να ισχύει ένας από τους 3 ελαχιστόρους. Μαζί δεν μπορεί να ισχύουν.

Ελαχιστόρος 0: A=0, B=0 και C=0, με άλλα λόγια το σήμα A =0, το σήμα B=0, και το C=0.

Πρέπει ο ελαχιστόρος αυτός να γραφτεί ως γινόμενο με τέτοιον τρόπο ώστε το γινόμενο αυτό να είναι 1, δηλαδή ο ελαχιστόρος να γραφτεί έτσι ώστε να δίνει 1 επειδή ισχύει ΜΟΝΟ αυτός σε κάποια χρονική στιγμή στην οποία το F=1
Άρα, γράφεται A’B΄C’:
Γιατί, αν A=0, A’=1.
Αν B=0, B’=1
Αν C=0, C’=1, άρα A’BC’=1 άρα F=1

ΠΩΣ θα γράψω ένα γινόμενο αλγεβρικό έτσι ώστε αυτό να δίνει 1 ενώ A=B=C=0;
Θα το γράψω A’B’C’

ΠΩΣ θα γράψω ένα γινόμενο αλγεβρικό έτσι ώστε αυτό να δίνει 1 ενώ A=B=0 ΚΑΙ C=1;
Θα το γράψω A’B’C

ΠΩΣ θα γράψω ένα γινόμενο αλγεβρικό έτσι ώστε αυτό να δίνει 1 ενώ A=B=C=1 ;
Θα το γράψω ABC

Άρα η **F= A’B’C’ + A’B’C + ABC**

Έστω A=0, B=0, C=0 (ελαχιστόρος 0)

F= 0’0’0’ + 0’0’0 + 000 =
1 + 0 + 0 = 1
(*ΕΝΑΣ ελαχιστόρος μόνο δίνει γινόμενο 1, εδώ ο 0*)

Έστω A=0, B=0, C=1 (ελαχιστόρος 1)

F= 0’0’1’ + 0’0’1 + 001 =
0 + 1 + 0 = 1
(*ΕΝΑΣ ελαχιστόρος μόνο δίνει γινόμενο 1, εδώ ο 1*)

Έστω A=1, B=1, C=1 (ελαχιστόρος 7)

F= 1’1’1’ + 1’1’1 + 111 =
0 + 0 + 1 = 1
(*ΕΝΑΣ ελαχιστόρος μόνο δίνει γινόμενο 1, εδώ ο 7*)

ΓΕΝΙΚΑ: Βρίσκουμε για ποιους ελαχιστόρους η **F=1**, γράφουμε το γινόμενο στην κατάλληλη μορφή (αν μία μεταβλητή του ελαχιστόρου έχει 0 στον πίνακα αληθείας γράφεται συμπληρωματικά, αλλιώς αν είναι 1 κανονικά). Τέλος αθροίζουμε τα επιμέρους γινόμενα.

lect5PG9

Έστω ένα κύκλωμα με 3 εισόδους το οποίο δίνει 1 όταν ισχύουν οι συνδυασμοί 3,5,6, και 7. Να βρείτε τη λογική συνάρτηση F σε μορφή αθροίσματος γινομένων, η οποία περιγράφει αυτό το κύκλωμα.

1ο βήμα: Κατασκευή πίνακα αληθείας

Έχουμε 3 εισόδους X, Y, Z έξοδο F

X	Y	Z	Ελαχιστόρος	F
0	0	0	0	0
0	0	1	1	0
0	1	0	2	0
0	1	1	3	1
1	0	0	4	0
1	0	1	5	1
1	1	0	6	1
1	1	1	7	1

2ο βήμα: Βρίσκουμε για ποιους συνδυασμούς η F=1 και γράφουμε κάθε συνδυασμό σε μορφή γινομένου όπου:

- Αν η αντίστοιχη μεταβλητή εισόδου στον πίνακα είναι 0, τότε γράφεται συμπληρωματικά, π.χ., X’
- Αν η αντίστοιχη μεταβλητή εισόδου στον πίνακα είναι 1, τότε γράφεται συμπληρωματικά, π.χ., X

Ελαχιστόρος 3: X’ Y Z
Ελαχιστόρος 5: X Y’ Z
Ελαχιστόρος 6: X Y Z’
Ελαχιστόρος 7: X Y Z

3ο βήμα: Κατασκευάζουμε το άθροισμα γινομένων. Αν ονομάσω τους ελαχιστόρους M3, M5, M6, M7, τότε η F περιγράφεται από το άθροισμα αυτών των M
Δηλαδή η F=1 όταν ισχύει **ΕΝΑΣ** από τους συνδυασμούς **M3, M5, M6, M7**. Σε κάθε χρονική στιγμή, οι είσοδοι βρίσκονται σε μία κατάσταση

Άρα οι είσοδοι κάθε χρονική στιγμή σχηματίζουν έναν από τους συνδυασμούς M0-M7.

Για τους M0, M1, M2, και M4 η F=0

Άρα η F είναι 1 όταν σχηματιστεί ο ελαχιστόρος M3 Ή ο M5 Ή ο M6 Ή ο M7

F= M3+ M5+ M6 + M7 = X’ Y Z + X Y’ Z + X Y Z’ + X Y Z

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΕΛΕΣΗΣ: Κάθε κύκλωμα σε μία χρονική στιγμή μπορεί να βρεθεί σε μία κατάσταση αναφορικά με τις εισόδους του. Άρα για να μελετήσουμε τη συμπεριφορά του πρέπει να μελετήσουμε καθεμία κατάσταση χωριστά.

Έστω τη *χρονική στιγμή t1* ότι X=0, Y=1, Z=1

```
F = 0’1 1 + 0 1’1 + 0 1 1’+ 0 1 1 =
    1 1 1 + 0 0 1 + 0 1 0 + 0 1 1 =
      1   +   0   +   0   +   0   = 1
```

Δηλαδή η F είναι 1 επειδή ο συνδυασμός M3 δίνει αποτέλεσμα 1. Οι συνδυασμοί M5, M6, M7, παρότι υπάρχουν στη λογική έκφραση της F δίνουν 0 για τιμές X=0, Y=1, Z=1

Έστω τη χρονική στιγμή *t2* ότι X=1, Y=1, Z=1
Τότε, από την **F= X’ Y Z + X Y’ Z + X Y Z’ + X Y Z** παίρνουμε αντικαθιστώντας τις τιμές:

```
F = 1’1 1 + 1 1’1 + 1 1 1’ + 1 1 1 =
    0 1 1 + 1 0 1 + 1 1 0  + 1 1 1 =
      0   +   0   +   0    +   1   = 1
```

Δηλαδή η F είναι 1 επειδή ο συνδυασμός M7 δίνει αποτέλεσμα 1. Οι συνδυασμοί M3, M5, M6, παρότι υπάρχουν στη λογική έκφραση της F δίνουν 0 για τιμές X=1, Y=1, Z=1

Έστω τη χρονική στιγμή *t3* ότι X=0, Y=0, Z=0
Τότε, από την **F= X’ Y Z + X Y’ Z + X Y Z’ + X Y Z** παίρνουμε αντικαθιστώντας τις τιμές:

```
F = 0’0 0 + 0 0’0 + 0 0 0’ + 0 0 0 =
    1 0 0 + 0 1 0 + 0 0 1  + 0 0 0 =
      0   +   0   +   0    +   0   = 0
```

Δηλαδή η F είναι 0 επειδή ο συνδυασμός M0 δίνει αποτέλεσμα 0. Δεν υπάρχει στο άθροισμα το M0

ΙΔΙΑ ανάλυση μπορείτε να κάνετε για καθέναν από τους M0-M7
Για τους M3,M5 ως M7 είναι F=1 (ΕΝΑΣ συνδυασμός θα είναι 1)
Για τους M0, M1,M2, και M4 θα είναι F=0 (ΚΑΝΕΝΑΣ συνδυασμός δεν θα δίνει 1, όλοι 0)

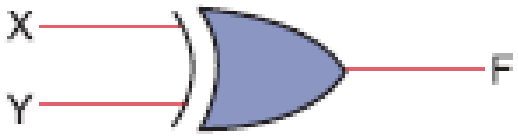
ΕΞΗΓΗΣΗ: γιατί κάθε γινόμενο γράφεται έτσι ώστε οι όροι που είναι 0 να εμφανίζονται συμπληρωματικοί ενώ εκείνοι που είναι 1 κανονικά;

Έστω ο όρος Μ3: Δηλαδή το X=0, Y=1, Z=1.
Το γράψαμε X’ Y Z πολύ απλά διότι θέλουμε για τον ΣΥΓΚΕΚΡΙΜΕΝΟ συνδυασμό τιμών εισόδου δηλαδή X=0, Y=1, Z=1 το γινόμενο να δίνει 1 έτσι ώστε η F=1. **Το γινόμενο θα είναι 1 όταν όλοι οι όροι που συμμετέχουν σε αυτό είναι 1.**

Άρα αφού το X=0 το γράφουμε X’ ώστε να γίνει 0’=1
Τα Y, Z είναι 1 άρα τα αφήνουμε ως έχουν.

lect5PG10

Αποκλειστικό ή (XOR)



Είσοδος		Έξοδος
X	Y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive OR.
Για 2 εισόδους: Δίνει 1 όταν οι δύο είσοδοι διαφέρουν μεταξύ τους ή διαφορετικά όταν υπάρχει μονός (περιττός) αριθμός μονάδων στις εισόδους.
F= X’ Y + X Y’

Για περισσότερες από 2 εισόδους: Μετράμε πόσοι είναι οι άσσοι και αν είναι άρτιο το πλήθος τους το αποτέλεσμα της F = 0 αλλιώς 1.

ΛΟΓΙΚΗ ΕΚΦΡΑΣΗ ΤΗΣ XOR

Βλέπουμε για ποιους συνδυασμούς η F = 1
Καθένας από αυτούς τους συνδυασμούς (ελαχιστόρους) γράφεται σε μορφή γινομένου όπου αν μία είσοδος είναι 1 γράφεται σε κανονική μορφή αν είναι 0 σε συμπληρωματική.
Αθροίζουμε τους ελαχιστόρους από το βήμα 2

1ο γινόμενο X’ Y
2ο γινόμενο X Y’
Άρα μία πύλη XOR περιγράφεται αλγεβρικά από τη σχέση:
F= X’ Y + X Y’

ΑΝΑΛΥΣΗ
Έστω τη χρονική στιγμή t1 ότι μία πύλη XOR δέχεται εισόδους X=0, Y=1

F= 0′ 1 + 0 1′ = 1 1 + 0 0 = 1

Έστω τη χρονική στιγμή t2 ότι μία πύλη XOR δέχεται εισόδους X=0, Y=0

F= 0′ 0 + 0 0′ = 1 0 + 0 1 = 0

Δηλαδή κάθε αλγεβρική έκφραση (γινόμενο) για έναν ελαχιστόρο γράφεται με τέτοιο τρόπο ώστε όταν κάποια στιγμή στις εισόδους του κυκλώματος εφαρμοστεί αυτός ο συνδυασμός το γινόμενο να είναι ΕΝΑ

Πχ, **η F δίνει 1 όταν X=1 Y=0**. Ο συνδυασμός αυτός **αλγεβρικά γράφεται σε μορφή γινομένου ως XY′** έτσι ώστε όταν εφαρμοστούν στις εισόδους οι τιμές X=1 Y=0 να δώσει 1.

lect5PG11

Αποκλειστικό ΟΥΤΕ (XNOR)

Είσοδος		Έξοδος
X	Y	F
0	0	1
0	1	0
1	0	0
1	1	1

F= X′ Y′ + X Y
Όταν οι είσοδοι είναι ίσες μεταξύ τους τότε η F=1

lect5PG13

Πλήρης Αθροιστής

Ενός bit (υλοποιεί μία στήλη του αθροίσματος)

Διασυνδεδεμένοι n αθροιστές παράγουν το άθροισμα αριθμών n Bit

Έστω ότι έχουμε 2 αριθμούς τεσσάρων bit

A = 0101
B = 1001

Για να τους προσθέσουμε, προσθέτουμε τα bit κάθε στήλης, δηλαδή Ai, Bi, και Ci (κρατούμενο προηγούμενης στήλης)
Το αποτέλεσμα που παράγεται από κάθε στήλη είναι ένα Bit αθροίσματος Si και ένα επόμενο κρατούμενο, Cnext

Στο παράδειγμα , όταν προστίθενται τα bit A0, B0 (εννοείται ότι C0=0), τότε S0= 1 + 1 =0 και Cnext = 1

ΣΥΜΠΕΡΑΣΜΑ: Ένα κύκλωμα που αναλαμβάνει να προσθέσει μία στήλη από bit, **δέχεται 3 εισόδους A,B και Cin** (*κρατούμενο εισόδου*)
Και **παράγει 2 εξόδους S, Cout** (*κρατούμενο εξόδου*)

Στο παράδειγμα από την πρόσθεση των στηλών A1 και B1 (A1=0, B1=0, C1=1)

0 + 0 + 1 θα λάβουμε S1=1, C2=0

Για να υλοποιήσουμε αθροιστή που προσθέτει αριθμούς μήκους n bit πρέπει να ενώσουμε n τέτοια κυκλώματα.

Επειδή έχει 3 εισόδους, θα έχουμε 2^3=8 συνδυασμούς εισόδων τις οποίες πρέπει να μελετήσουμε ξεχωριστά.

_____ **2ο γκρουπ** _____

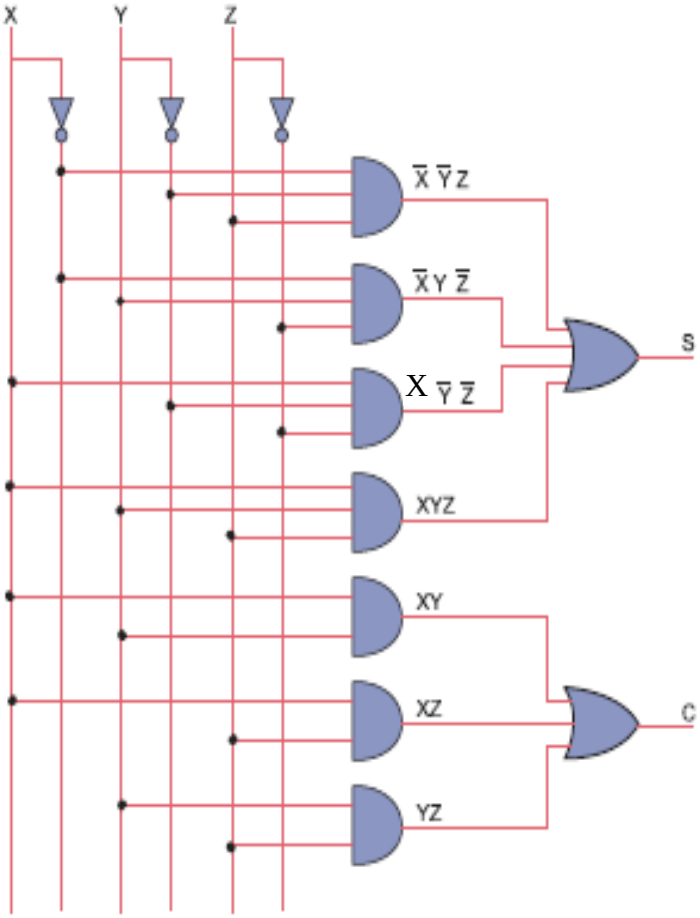
Πχ A = 1010
B = 0101

1010 +
0101 =
0
_____

Αν θέλουμε να προσθέσουμε αριθμούς N bit χρειάζονται N αθροιστές ενός bit
Έστω η στήλη 0 (δεξιότερη).
Τα στοιχεία που προστίθενται είναι A0, B0, και το C0 (το κύκλωμα έχει 3 εισόδους, οι οποίες σχηματίζουν 2^3 = 8 συνδυασμούς).
Παράγεται ένα bit αθροίσματος S0 και ένα νέο κρατούμενο το C1 (το κύκλωμα έχει 2 εξόδους).
Άρα μιλάμε για κύκλωμα με 3 εισόδους και 2 εξόδους

lect5PG14

ΛΟΓΙΚΟ ΔΙΑΓΡΑΜΜΑ Αθροιστή 1 Bit (ΜΙΑΣ ΣΤΗΛΗΣ)



Κατασκευή του πίνακα αληθείας με στόχο να μετατρέψουμε τις αριθμητικές πράξεις σε λογικές.

ΕΙΣΟΔΟΙ				**ΕΞΟΔΟΙ**	
X	**Y**	**Cin**	**Ελαχιστόρος**	**S**	**Cout**
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	2	1	0
0	1	1	3	0	1
1	0	0	4	1	0
1	0	1	5	0	1
1	1	0	6	0	1
1	1	1	7	1	1

Εργαζόμαστε όπως γνωρίζουμε για κάθε έξοδο ξεχωριστά

S= X’ Y’ Cin + X’ Y Cin’ + X Y’ C’in + X Y Cin (*Τα 4 γινόμενα αντιστοιχούν στους ελαχιστόρους 1,2,4,7*). Συμβολικά γράφεται **S= Σ(1,2,4,7)**, δηλαδή το S είναι άθροισμα των ελαχιστόρων 1, 2, 4, και 7 όπου κάθε ελαχιστόρος είναι ένα γινόμενο.
Cout= X’ Y Cin + X Y’ Cin + X Y C’in + X Y Cin. Συμβολικά: **Cout= Σ(3, 5,6,7)**

Εναλλακτικά, επειδή το S = 1 όταν το πλήθος των μονάδων είναι περιττό, **S = X xor Y xor Cin**. Δηλαδή τα σήματα X, Y, Z θα συνδέονταν σε μία XOR τριών εισόδων, της οποίας η έξοδος θα ήταν το S.

Έξοδος Cout

Cout= X’ Y Cin + X Y’ Cin + X Y C’in + X Y Cin

Επειδή γενικά ισχύει ότι **X+X=X**, *έχω το δικαίωμα στην παραπάνω σχέση να προσθέσω άλλες 2 φορές το γινόμενο X Y Cin*

Cout= X’ Y Cin+ X Y Cin + X Y’ Cin + X Y Cin + XY Cin’ + X Y Cin

Παίρνω τα γινόμενα ανά δύο

1ο γινόμενο + 2ο γινόμενο: X’ Y Cin+ X Y Cin = Y Cin (X+X’) = Y Cin (επειδή X + X’ =1)
3ο γινόμενο + 4ο γινόμενο: X Y’ Cin+ X Y Cin = X Cin (Y+Y’) = X Cin (επειδή Y + Y’ =1)
5ο γινόμενο + 6ο γινόμενο: X Y C’in+ X Y Cin = X Y (Cin+C’in) = X Y (επειδή Cin + Cin’ =1)

Cout = XY + XCin + Y Cin

X + X’ =1 γιατί αν X=0 έχουμε 0+1 =1 ενώ αν X=1 έχουμε 1+0=1
X+X = X γιατί αν X=0 έχουμε 0+0=0, αν X=1 έχουμε 1+1=1

ΒΗΜΑΤΑ ΣΧΕΔΙΑΣΗΣ

- Γράφω κατακόρυφα τα σήματα εισόδου
- Δίπλα από κάθε σήμα σχεδιάζω έναν αντιστροφέα για να πάρω το συμπληρωματικό
- Για κάθε γινόμενο που έχω βγάλει από τον πίνακα αληθείας κάνω τις κατάλληλες συνδέσεις (τελίτσες) και τις ενώνω σε μία AND
Όλα τα γινόμενα σε μία OR

X = bit του πρώτου αριθμού
Y = bit του δεύτερου αριθμού
Z = κρατούμενο εισόδου (προηγουμένως το συμβολίσαμε ως Cin)
S = bit αθροίσματος
C = κρατούμενο εξόδου (προηγουμένως το συμβολίσαμε ως Cout)

3 είσοδοι άρα 8 συνδυασμοί εισόδων (ελαχιστόροι)

X	Y	Z	Ελαχιστόρος	S	C
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	2	1	0
0	1	1	3	0	1
1	0	0	4	1	0
1	0	1	5	0	1
1	1	0	6	0	1
1	1	1	7	1	1

S= X'Y'Z + X'YZ' + XY'Z' + XYZ (1)
C= X'YZ + XY'Z + XYZ' + XYZ (2)

Π.χ. έστω τη χρονική στιγμή t1 ο αθροιστής δέχεται ως εισόδους (δηλαδή μία στήλη) τις X=0, Y=0, Z=1

S = 0' 0' 1 + 0'0 1' + 0 0'1'+ 0 0 1 =
1 1 1 + 1 0 0 + 0 1 0 + 0 0 1 =
1 + 0 + 0 + 0 = 1

Δηλαδή ο πρώτος όρος γινομένου (**ΚΑΙ ΜΟΝΟ αυτός**) είναι 1 για τον συνδυασμό εισόδων 0, 0, 1 και κάνει την S=1

C = 0' 0 1 + 0 0' 1 + 0 0 1' + 0 0 1
= 0 + 0 + 0 + 0 = 0

ΠΑΡΑΤΗΡΗΣΕΙΣ

Αν κάποιος προσέξει από τον πίνακα αληθείας ότι το S=1 όταν το πλήθος μονάδων στις εισόδους είναι περιττό, τότε μπορεί να πει ότι **S= X xor Y xor Z** (3). Η (1) είναι ισοδύναμη της (3). Δηλαδή για κάθε συνδυασμό εισόδων τα αποτελέσματά τους είναι ίδια.

2) X OR X = X, X OR X OR X= X. Επίσης X OR X'=1 (γιατί, αν X=0, 0+1=1, αν X=1, 1+0=1)

C= X' Y Z + X Y' Z + X Y Z' + XYZ

Έχω δικαίωμα στην C να προσθέσω άλλες 2 φορές το γινόμενο XYZ (δεν αλλάζει το αποτέλεσμα αφού XYZ+XYZ+XYZ= XYZ)

C= [X' Y Z + XYZ]+ [X Y' Z + XYZ] + [X Y Z' + XYZ]
(άθροισμα 6 γινομένων)

Παίρνω τα γινόμενα ανά 2:

X' Y Z + XYZ = YZ (X' + X) = YZ
X Y' Z + XYZ = XZ (Y' + Y) = XZ
X Y Z'+ XYZ = XY (Z + Z') = XY

Τελικά ***C= XY+XZ+YZ***

Η ιδέα είναι ότι καθένα από τα 3 πρώτα γινόμενα του C έχει ένα συμπληρωματικό όρο (μία το X μία το Y και μία το Z). Άρα αν «κολλήσω» σε καθένα από αυτά το XYZ μπορώ να πετύχω απαλοιφές.

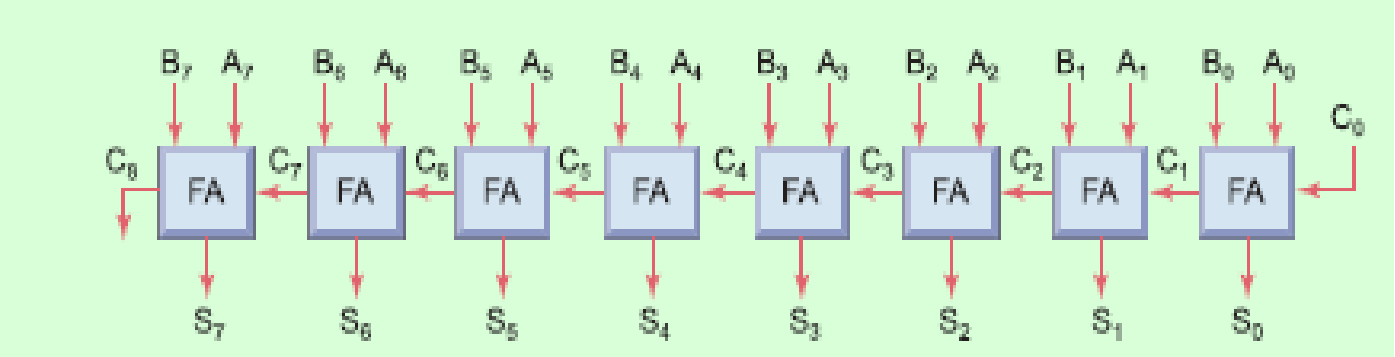
ΒΗΜΑΤΑ ΣΧΕΔΙΑΣΗΣ

1. Γράφω κατακόρυφα τα σήματα εισόδου
2. Δίπλα από κάθε σήμα σχεδιάζω έναν αντιστροφέα για να πάρω το συμπληρωματικό
3. Για κάθε γινόμενο που έχω βγάλει από τον πίνακα αληθείας κάνω τις κατάλληλες συνδέσεις (τελίτσες) και τις ενώνω σε μία AND
4. Όλα τα γινόμενα σε μία OR

S= X′ Y′ Z + X′ Y Z′ + X Y′ Z′ + X Y Z
C= XY+XZ+YZ
S= X XOR Y XOR Z (Δεν θα υπήρχαν AND και η XOR τριών εισόδων θα συνδεόταν με τα σήματα X, Y Z)

lect5PG15

ΑΘΡΟΙΣΤΕΣ n bit



FA = το κύκλωμα πλήρους αθροιστή ενός bit που είδαμε προηγουμένως (Full Adder). Για να υλοποιήσουμε αθροιστή 8 bit συνδέουμε 8 τέτοιους αθροιστές.

ΠΑΡΑΔΕΙΓΜΑ
Π.χ έστω 2 μη προσημασμένοι αριθμοί, A = 24 και B= 100 εκφρασμένοι σε ένα byte ο καθένας. Να δείξετε την πρόσθεση πάνω στο κύκλωμα αθροιστή 8 bit

A = (A7A6A5A4A3A2A1A0) = 00011000
B = (B7B6B5B4B3B2B1B0) = 01100100

```
   00011000 +
   01100100
          0
_______________
```

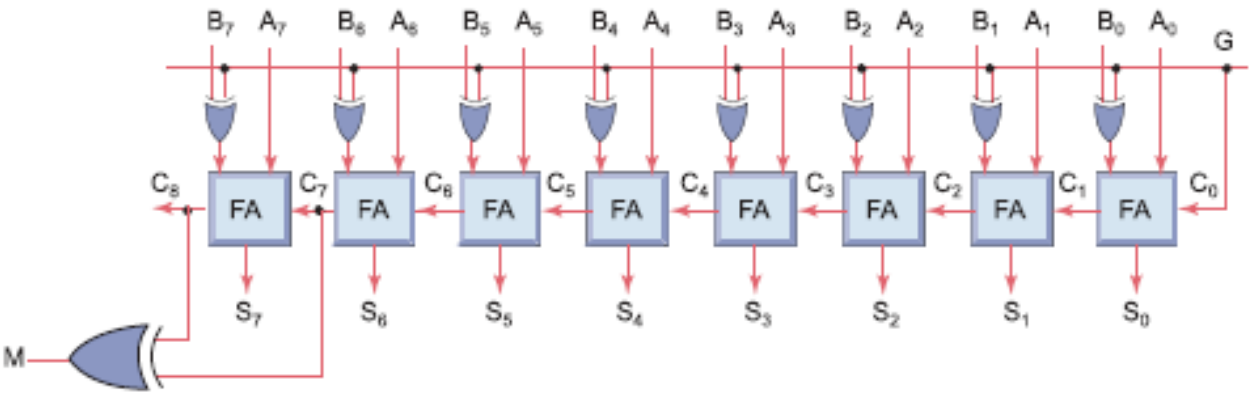
Ξεκινώντας από δεξιά:

Στήλη 0: A0 = 0, B0=0, C0=0, άρα S0=0 και το C1=0. Το C1 είναι το κρατούμενο εξόδου του πρώτου αθροιστή από δεξιά και ταυτόχρονα είσοδος στον επόμενο αριστερότερο αθροιστή. Γενικά, κάθε αθροιστής παράγει ως έξοδο ένα κρατούμενο με το οποίο τροφοδοτεί τον αμέσως αριστερότερό του.
Στήλη 1: A1 = 0, B1=0, C1=0, άρα S1=0 και το C2=0.
Στήλη 2: A2 = 0, B2=1, C2=0, άρα S2=1 και το C3=0.
Στήλη 3: A3 = 1, B3=0, C3=0, άρα S3=1 και το C4=0.
Στήλη 4: A4 = 1, B4=0, C4=0, άρα S4=1 και το C5=0.
Στήλη 5: A5 = 0, B5=1, C5=0, άρα S5=1 και το C6=0.
Στήλη 6: A6 = 0, B6=1, C6=0, άρα S6=1 και το C7=0.
Στήλη 7: A7 = 0, B7=0, C7=0, άρα S7=0 και το C8=0.

Άρα το άθροισμα είναι S7S6S5S4…..S0= 01111100 = 124

lect5PG16

Αθροιστές-Αφαιρέτες με έλεγχο προσήμου



A-B:

Τα σήματα A τροφοδοτούν τον αθροιστή στην κανονική τους μορφή.
Τα **B περνούν από μία XOR** και ειδικότερα κάθε bit Bi λαμβάνει μέρος σε μία πράξη XOR μαζί με ένα σήμα G
Δηλαδή έχουμε *B0 XOR G, B1 XOR G, …….. B7 XOR G*

2 περιπτώσεις για το σήμα G:
A) G=0. Αν το G=0 τότε στις πύλες XOR Εκτελείται η πράξη **Bi XOR 0.** Έχουμε 2 υπο-περιπτώσεις
A1) Bi =0, τότε η πράξη XOR θα δώσει **0 XOR 0 = 0**
A2) Bi =1, η πράξη XOR θα δώσει **1 XOR 0 = 1**
Από τις περιπτώσεις A1, A2 συμπεραίνουμε ότι το αποτέλεσμα των XOR είναι ότι είναι και το Bi. Συνεχίζοντας, αυτό σημαίνει ότι οι αθροιστές FA θα εκτελούν την πράξη **Ai + Bi + C0 = Ai+ Bi**, δηλαδή **πρόσθεση**. Στην πρόσθεση το αρχικό κρατούμενο **C0 = 0**

Τελικά, το κύκλωμα λειτουργεί ως αθροιστής

Β) G=1, από τις πύλες XOR θα εκτελεστεί η πράξη **Bi XOR 1**.
Β1) Bi=0 άρα εκτελείται η **0 XOR 1 = 1**
Β2) Bi=1, άρα **1 XOR 1 = 0**
Από Β1 και Β2 συνεπάγεται ότι το αποτέλεσμα της XOR είναι πάντοτε αντίστροφο του Bi, δηλαδή B'i
ΔΗΛΑΔΗ B XOR 1 = B'

Άρα τι δέχεται ως είσοδο ο αθροιστής;
ΔΕΝ βάζω δείκτες το γράφω γενικά.
A + B' + 1 (δηλαδή το C0=1)

A + B' +1 *= A + Σ1(B) +1 = A + Σ2(B) =* ***A-B***
Άρα, αν η πράξη που πρέπει να εκτελεστεί είναι αφαίρεση, τότε η ALU στέλνει ένα σήμα ίσο με 1 στο G. Αυτό μετατρέπει αυτομάτως τον αθροιστή σε αφαιρέτη.

ΥΠΕΡΧΕΙΛΙΣΗ: Για να ελέγξουμε την υπερχείλιση ελέγχουμε αν τα bit C8 και C7 είναι ίδια. **Αν C8=C7 τότε M=0, ενώ αν C7 <> C8 τότε η XOR δίνει 1 δηλαδή M=1**.

Έχουμε **υπερχείλιση όταν τα C7 και C8 είναι διαφορετικά**. Άρα αν το M = 1 τότε αυτό το σήμα δίνεται πίσω σε έναν καταχωρητή (κύκλωμα της CPU που αποθηκεύει δεδομένα τα οποία η CPU επεξεργάζεται) της CPU. Αυτό είναι ένα σήμα ειδοποίησης για Overflow, το οποίο εφόσον είναι 1 σημαίνει ότι θα δοθεί μήνυμα λάθους.

Έστω προσθέτουμε 2 θετικούς αριθμούς ενός byte

A7=B7=0 αν το S7=1 τότε ο αριθμός που βγαίνει ως αποτέλεσμα είναι αρνητικός!
Για να συμβεί αυτό πρέπει C7=1
A7+B7+C7=0+0+C7=1. Σε αυτή την περίπτωση το C8=0
Άρα C7<>C8. *Άρα περνώντας τα C7, C8 από την XOR θα έχουμε M=1* (σήμα υπερχείλισης)

_____ 2ο γκρουπ _____

Κάθε αθροιστής FA ενός bit δέχεται ως είσοδο τα bit του αριθμού A και τα bit του B xor G
Το G συνδέεται με το κρατούμενο εισόδου C0

Στις XOR εκτελείται η πράξη G XOR Bi. Υπάρχουν 2 περιπτώσεις:
1) G=0
1.1) Έστω ότι ένα τυχαίο bit **Bi =0**. τότε εκτελείται η πράξη **Bi XOR G = 0 XOR 0 = 0 = Bi**
1.2) Έστω ότι ένα τυχαίο bit **Bi =1**. τότε εκτελείται η πράξη **Bi XOR G= 1 XOR 0 =1 = Bi**
Από τις 1.1 και 1.2 καταλήγουμε ότι **όταν G=0, H XOR δίνει αποτέλεσμα ότι είναι το Bit Bi**
Αυτό σημαίνει ότι **από τις XOR περνάνε τα B**. Άρα το κύκλωμα εκτελεί την πράξη: **Ai + Bi + G = Ai+ Bi +0 = Ai+Bi**. Άρα, **όταν G=0 το κύκλωμα κάνει πρόσθεση** δηλαδή είναι αθροιστής. Με άλλα λόγια,

αν G=0 τα Β εισέρχονται στον αθροιστή ΑΥΤΟΥΣΙΑ. Τα Α εισέρχονται στον αθροιστή ΑΥΤΟΥΣΙΑ.

2) G=1

2.1) Έστω ότι ένα τυχαίο bit **Bi =0**. τότε εκτελείται η πράξη **Bi XOR G = 0 XOR 1 = 1 = B'i**

2.2) Έστω ότι ένα τυχαίο bit **Bi =1**. τότε εκτελείται η πράξη **Bi XOR G= 1 XOR 1 =0 = B'i**

Από τις 2.1 και 2.2 καταλήγουμε ότι **όταν G=1, Η XOR δίνει αποτέλεσμα το συμπλήρωμα του Bi**

ΠΟΙΑ ΠΡΑΞΗ ΕΚΤΕΛΕΙΤΑΙ;
Ai + B'i + G = Ai + [B'i + 1] = Ai + [Σ1(Βι) +1] = Αι + Σ2 (Βι) = A-B
Άρα όταν G=1 το κύκλωμα κάνει αφαίρεση

Όταν στη CPU (ειδικότερα στην ALU) έρθει εντολή πρόσθεσης, **ο αθροιστής δέχεται ένα σήμα G=0, όταν έρθει εντολή αφαίρεσης G=1**

Έλεγχος Υπερχείλισης: Αν C7<> C8 τότε έχουμε υπερχείλιση.
Περνώντας τα C7, C8 από μία XOR αυτή θα δώσει σήμα M=1 αν τα δύο κρατούμενα είναι διαφορετικά. ΄Άρα όταν ο αθροιστής δώσει σήμα M =1 πίσω στην CPU λαμβάνουμε σήμα OVERFLOW.

ΠΑΡΑΔΕΙΓΜΑ
Έστω προσθέτουμε 2 θετικούς αριθμούς ενός byte

A7=B7=0 αν το S7=1 τότε ο αριθμός που βγαίνει ως αποτέλεσμα είναι αρνητικός!
Για να συμβεί αυτό πρέπει C7=1
A7+B7+C7=0+0+C7=1. Σε αυτή την περίπτωση το C8=0
Άρα C7<>C8. Άρα περνώντας τα C7, C8 από την XOR θα έχουμε M=1 (σήμα υπερχείλισης

ΑΣΚΗΣΗ

- Ένα κύκλωμα σχηματίζει στις εισόδους του τους αριθμούς 0-15 και οι έξοδοι σχηματίζουν την τετραγωνική ρίζα των αριθμών της εισόδου, αν αυτή η ρίζα είναι ακέραια. Διαφορετικά, σχηματίζουν τον αριθμό 0.
- Πόσες εισόδους και εξόδους έχει το κύκλωμα;
- Να βρείτε τις λογικές εκφράσεις των εξόδων και να σχεδιάσετε το κύκλωμα

ΛΥΣΗ

Α) Για να σχηματιστούν στην είσοδο οι αριθμοί 0-15 θέλουμε 4 εισόδους.
Β) Ο μικρότερος αριθμός 0 ο μεγαλύτερος το 15. Η μεγαλύτερη ακέραια ρίζα αφορά την είσοδο 9 και είναι το 3

'Αρα οι έξοδοι είναι (ρίζες από 0-3) ΔΥΟ, αφού με 2 Bit μπορώ να σχηματίσω τους αριθμούς 0–3

A	B	C	D	F1	F0	**Παρατηρήσεις**
0	0	0	0	0	0	
0	**0**	**0**	**1**	0	1	(ρίζα 1= 1)
0	0	1	0	0	0	
0	0	1	1	0	0	
0	**1**	**0**	**0**	1	0	(ρίζα 4= 2)
0	1	0	1	0	0	
0	1	1	0	0	0	
0	1	1	1	0	0	
1	0	0	0	0	0	
1	**0**	**0**	**1**	1	1	(ρίζα 9= 3)
1	0	1	0	0	0	
1	0	1	1	0	0	
1	1	0	0	0	0	
1	1	0	1	0	0	
1	1	1	0	0	0	
1	1	1	1	0	0	

F1= A′ B C′ D′ + **A B′ C′ D** (**ΣΧΕΔΙΑΖΕΤΑΙ ΜΙΑ ΦΟΡΑ ΣΤΟ ΣΧΗΜΑ**)
F0 = A′ B′ C′ D + **A B′ C′ D**

$= A'BC'D' + AB'C'D$

$F0 = A'BC'D' + AB'C'D$