Εισαγωγή στην Πληροφορική και τον Προγραμματισμό Η/Υ

4ο Μάθημα

Bit-Byte-word. Πρόσθεση / Αφαίρεση

Λεωνίδας Αλεξόπουλος
Αν. Καθηγητής ΕΜΠ
E-mail: leo@mail.ntua.gr
Τηλ: 210 772-1666

Περίληψη Μαθήματος

Στα προηγούμενα μαθήματα:

1- Τι είναι Η/Υ

2- Κωδικοποίηση-Επεξεργασία- Αποκωδικοποίηση

3- Συστήματα Αρίθμησης: 2δικό, 8δικό, 10δικό, 16δικό

4- Μετατροπή από το ένα σύστημα στο άλλο

Σήμερα:

5- bit/byte/word

6- Η πρόσθεση στο Δυαδικό

7- Παράσταση Αρνητικών

8- Η αφαίρεση στο Δυαδικό

Προγραμματισμός σε Η/Υ

Εφαρμογές

Γλώσσες Προγραμματισμού

```
function rowTotals = rowsum
% Add the values in each row and
% store them in a new array.

x = ones(2,10);
[n, m] = size(x);
rowTotals = zeros(1,n);
for j = 1:n
    Press Shift+Enter to rename 5 instances of 'i' to 'j'
end

    function colsum = addToSum
        colsum = 0;
        thisrow = x(i,:);
        for i = 1:m
            colsum = colsum + thisrow(i);
        end
    end

end
```

Λειτουργικό Σύστημα

Assembly Language	Machine Code
add $t1, t2, $t3	04CB: 0000 0100 1100 1011
addi $t2, $t3, 60	16BC: 0001 0110 1011 1100
and $t3, $t1, $t2	0299: 0000 0010 1001 1001
andi $t3, $t1, 5	22C5: 0010 0010 1100 0101
beq $t1, $t2, 4	3444: 0011 0100 0100 0100
bne $t1, $t2, 4	4444: 0100 0100 0100 0100
j 0x50	F032: 1111 0000 0011 0010
lw $t1, 16($s1)	5A50: 0101 1010 0101 0000
nop	0005: 0000 0000 0000 0101
nor $t3, $t1, $t2	029E: 0000 0010 1001 1110
or $t3, $t1, $t2	029A: 0000 0010 1001 1010
ori $t3, $t1, 10	62CA: 0110 0010 1100 1010
ssl $t2, $t1, 2	0455: 0000 0100 0101 0101
srl $t2, $t1, 1	0457: 0000 0100 0101 0111
sw $t1, 16($t0)	7050: 0111 0000 0101 0000
sub $t2, $t1, $t0	0214: 0000 0010 0001 0100

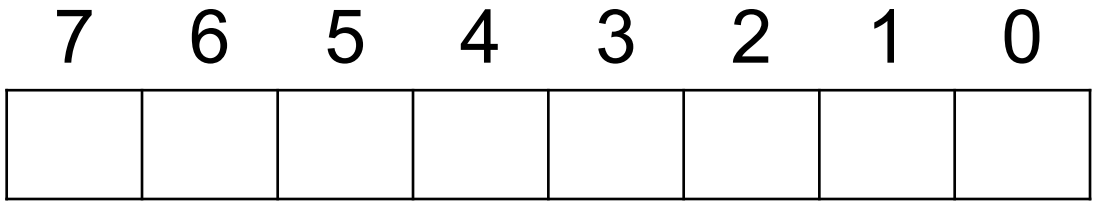
Γλώσσα Μηχανής

Μικρολειτουργίες & Μικροπρογραμματισμός

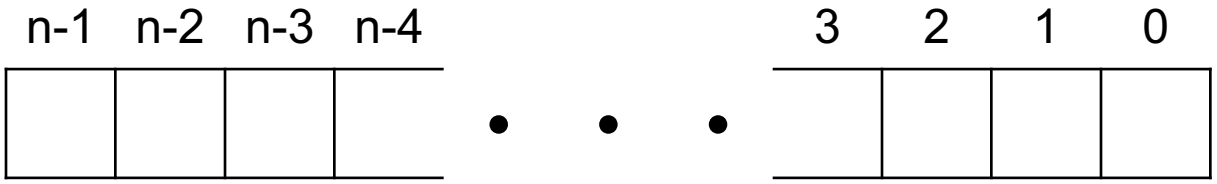
Ψηφιακή Λογική

bit – byte - word

- Κάθε ψηφίο {0,1} ενός δυαδικού αριθμού (π.χ 11011_2) είναι ένα δυαδικό ψηφίο (**BI**nary digi**T**} δηλ. **bit**
- Μία ομάδα **8 bit** αποτελεί ένα **byte(B)**



- Τα **bit** που μπορεί να επεξεργασθεί ένας Η/Υ σε ένα κύκλο λειτουργίας του λέγεται **ψηφιολέξη (word).** Ο αντίστοιχος αριθμός είναι το **μήκος ψηφιολέξης (wordlength)**

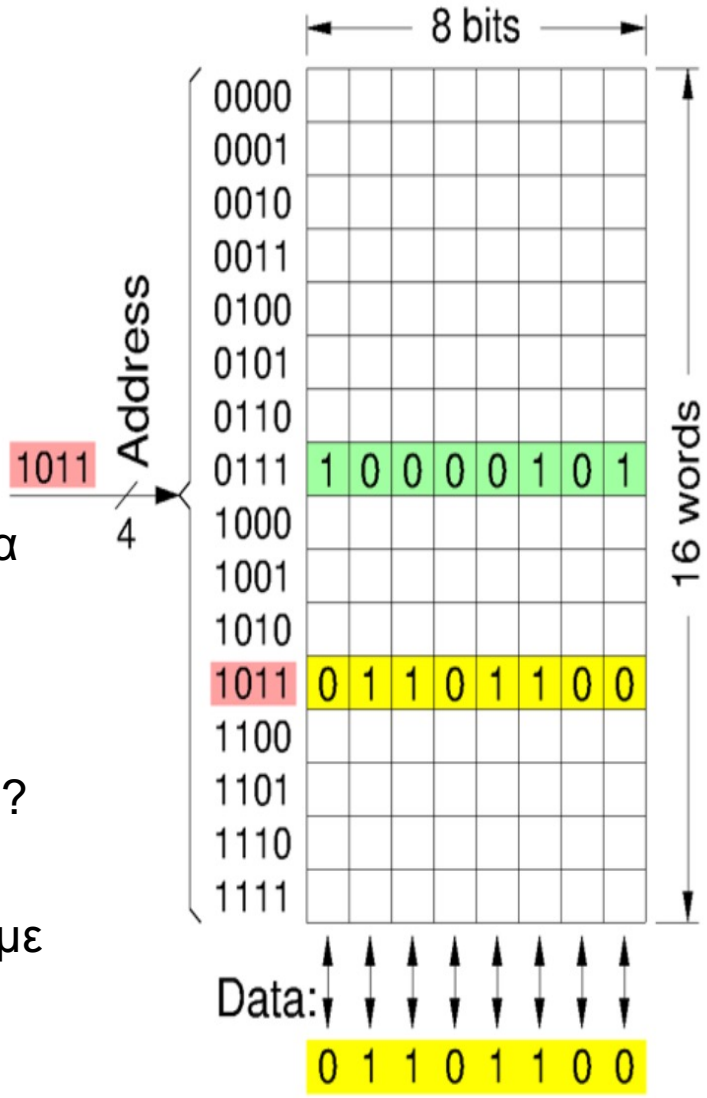



1-byte (8bit) in an electromechanical programmable computer, the Z3, 1941

bit – byte - word

Χρήση Bit/byte/word στην μνήμη του Η/Υ

- Κάθε οριζόντια γραμμή (όροφος) είναι μία λέξη (word)
- Μήκος ψηφιολέξης: 8 bits = 1byte
- Στο διπλανό σχήμα υπάρχουν 16 λέξεις η μία κάτω από την άλλη από θέση 0 σε θέση 15
- Μπορώ να επεξεργασθώ μία λέξη την φορά (σε ένα κύκλο λειτουργίας)
- Θέλω να χρησιμοποιήσω τον αριθμό στην θέση 11 (→ 1011)
- Πόσο είναι η χωρητικότητα της διπλανής «μνήμης»?
- 16 bytes
- Το πλήθος των οριζόντιων γραμμών περιγράφεται με δυνάμεις του δύο….
- Περισσότερα για μνήμη Η/Υ θα πούμε σε άλλα μαθήματα



Ref: https://www.csd.uoc.gr/~hy120/index.html

kB, MB, GB, TB …

- 10^3 Bytes → 1kB (SI & IEC, IEEE, EU, ISO, NIST)
- 10^6 Bytes → 1MB - Mega
- 10^9 Bytes → 1GB - Giga
- 10^12 Bytes → 1TB - Tera
- … Peta(15), Exa(18), Zetta(21), Yotta(24)

Αλλά παραδοσιακά χρησιμοποιούμε βάση το δύο:

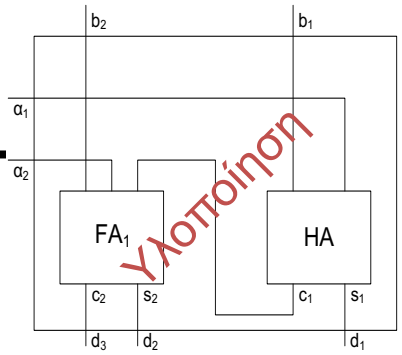
- 2^10 Bytes → 1kB = 1024 Bytes (π.χ computer RAM)
- 2^20 Bytes → 1MB = 1024 kBytes
- 2^30 Bytes → 1GB = 1024 MBytes
- 2^40 Bytes → 1TB = 1024 GBytes

IEC (International Electrotechnical Commission) & IEEE, EU, ISO, NIST προτείνουν την χρήση τη λέξης Kibi-, Mebi-, Gibi-, Tebi-, Pebi- για το δυαδικό σύστημα αλλά *χωρίς* ευρεία χρήση….

Πρόσθεση Δυαδικών Ακεραίων

- 0+0=0 0+1=1 1+0=1 1+1=10
- 4 βασικά παραδείγματα

	1	111	11	←κρατούμενο
101	1001	1011	1011	
10+	101+	101+	11+	
111	1110	10000	1110	



- Έστω Η/Υ με ψηφιολέξη=1byte

	1100 0100	(196)
	1001 1000	+ (152)
overflow digit→	**1**0101 1100	>255

Αφαίρεση Δυαδικών Ακεραίων

- 0-0=0 1-0=1 1-1=0 10-01=01 ← με δανεισμό από ανώτερη τάξη

- **Πχ**

$$\begin{array}{r} 1\,1\,0\,1 \\ 1\,1\,0\,- \\ \hline 1\,1\,1 \end{array}$$

⇒ **η αφαίρεση απαιτεί «λογική»!**

- Το “-6” πως το γράφουμε στο δυαδικό σύστημα?
- **Βασική Ιδέα**: 13-6=13+(-6) δηλ. η αφαίρεση μπορεί γίνει μέσω πρόσθεσης
- **Άρα**: ζητούμε μοντέλο παράστασης αριθμών που να:
  - κάνει δυνατή τη παράσταση αρνητικών αριθμών, και
  - είναι τέτοιο ώστε η αφαίρεση να γίνεται άμεσα χωρίς επιπρόσθετη λογική

Παράσταση Αρνητικών

→ *Πρόσημο-μέτρο (sign and magnitude code)*

→ *Συμπλήρωμα ως προς 2 (2-complement code)*

→ *Πόλωση - Πλεονασμός κατά Κ*

→ *IEEE*

Παράσταση Αρνητικών: πρόσημο-μέτρο (sign and magnitude code)

- **Σύμβαση**: το **MSD** δηλώνει το πρόσημο (π.χ. **0** : θετικό, **1** : αρνητικό).
  Παραδ.: **+**13_{10} = **0**1101_2 , **-**13_{10} = **1**1101_2.
- **Παρατηρήστε**: 5_{10} = **0**0101_2 τότε 5-13=5+(**-**13)

  (**-**13) **1** 1 1 0 1
  (**+**5) **0** 0 1 0 1 +
  $-8_{10} \neq$ 1 0 0 0 1 0_2

- **Λάθος !** Αυτή παράσταση, δεν καλύπτει την απαίτηση: «*η αφαίρεση να μπορεί γίνει μέσω πρόσθεσης*»

Παράσταση Αρνητικών: Συμπλήρωμα ως προς 2 (2-complement code)

- **Σύμβαση**: Για μήκος λέξης n-bit, το MSD εκφράζει το -2^{n-1}. Έτσι εκφράζονται οι αριθμοί από $(2^{n-1}-1)$ έως -2^{n-1} π.χ. Αν n=6 τότε εκφράζονται οι αριθμοί από –32 έως 31.

-32 16 8 4 2 1 Ποια «τάξη» παριστάνει κάθε στήλη

0 1 1 1 1 1 = 16+8+4+2+1=31. Πως θα αναπαραστήσουμε το «2»?

? ? ? ? ? ?

0 0 0 0 1 0 = 2 . Πως είναι το μηδέν σε C2?

? ? ? ? ? ?

0 0 0 0 0 0 = 0 . Είναι το 111111 ο μέγιστος?

1 1 1 1 1 1 = -32+16+8+4+2+1=-1 . Ποιος είναι ο MIN & MAX για n=6?

1 0 0 0 0 0 = -32 Το -32 είναι το MIN

0 1 1 1 1 1 =+31 Το +31 είναι το MAX. Είναι αντίστροφοι (NOT)

Ας δούμε τώρα δύο άλλους αντίστροφους:

0 0 1 1 1 0 = 8+4+2=14. Σε ποιον αριθμό το 10δικό αντιστοιχεί ο αντίστροφος του 14?

1 1 0 0 0 1 = -32+16+1 = -15

1 1 1 1 1 1 = -32+16+8+4+2+1=-1

οπότε πως μπορούμε να αναπαραστήσουμε ένα αρνητικό αριθμό a (πχ -14) στο C2?

Παράσταση Αρνητικών: Συμπλήρωμα ως προς 2 (2-complement code)

- Για ένα ακέραιο ***α***, για να βρούμε το συμπλήρωμα του ως προς 2, $c_2(\alpha)$: Κάνουμε αντιστροφή των bits της δυαδικής παράστασης του ***α***, και προσθέτουμε σε αυτό 1 π.χ. ***α*** =43 και n=7

-64	**32**	**16**	**8**	**4**	**2**	**1**	
0	1	0	1	0	1	1	= 43
1	0	1	0	1	0	0	
				+		1	
1	0	1	0	1	0	1	= $c_2(43)$ = -43

Η Αφαίρεση με την Παράσταση Συμπληρώματος ως προς 2

Παραδείγματα: Αφαίρεση σε 6-μπιτο Η/Υ

5-18 = 5+ (-18)

	-32	16	8	4	2	1	
	0	1	0	0	1	0	= 18
	1	0	1	1	0	1	
+						1	
	1	0	1	1	1	0	= c_2(18)
+	0	0	0	1	0	1	= 5
-32+16+2+1 =	1	1	0	0	1	1	=-13

0 0 1 1 0 0

(κρατούμενα κατά την πρόσθεση)

Οι παραπάνω πράξεις είναι σωστές γιατί το κρατούμενο που προστίθεται στο MSD είναι το ίδιο με το κρατούμενο που ‘εξέρχεται’ από το MSD.

Η Αφαίρεση με την Παράσταση Συμπληρώματος ως προς 2

Παραδείγματα: Αφαίρεση σε 6-μπιτο Η/Υ

29 + (-7). Γράψτε το «7»

-32	16	8	4	2	1	
0	0	0	1	1	1	=7 → NOT
1	1	1	0	0	0	→ +1
+					1	
1	1	1	0	0	1	= c_2(7)→+29
+ 0	1	1	1	0	1	= 29
0	1	0	1	1	0	= 22
1	1	1	0	0	1	

(κρατούμενα κατά την πρόσθεση)

Οι παραπάνω πράξεις είναι σωστές γιατί το κρατούμενο που προστίθεται στο MSD είναι το ίδιο με το κρατούμενο που 'εξέρχεται' από το MSD. Το αντίθετο, σημαίνει ότι για το συγκεκριμένο εύρος λέξης n bit, δεν μπορούμε να αναπαραστήσουμε το άθροισμα. Όταν τα ψηφία δεν φτάνουν για να χειρισθούμε τον αριθμό λέμε ότι έχουμε είτε **Overflow (*υπερχείλιση*),** είτε **Underflow («*υποχείλιση*»).**

Η Αφαίρεση με την Παράσταση Συμπληρώματος ως προς 2:

Υπερχείλιση & Υποχείλιση

Overflow: αποτέλεσμα $>(2^{n-1}-1)$

π.χ. n=6, 22 + 22 = 44

-32	16	8	4	2	1	
0	1	0	1	1	0	= 22
+ 0	1	0	1	1	0	= 22
1	0	1	1	0	0	=-20≠44
0	1	0	1	1	0	

Underflow: αποτέλεσμα $<-2^{n-1}$

π.χ. n=6, -8-31=(-8)+(-31)=-39

-32	16	8	4	2	1	
1	1	1	0	0	0	= -8
+ 1	0	0	0	0	1	= -31
0	1	1	0	0	1	=25≠-39
1	0	0	0	0	0	

Διαπίστωση : με τη παράσταση συμπληρώματος ως προς 2

- γίνεται δυνατή η παράσταση αρνητικών αριθμών,
- η αφαίρεση γίνεται άμεσα, μέσω πρόσθεσης, και χωρίς επιπρόσθετη λογική, και
- παρέχει άμεσο μηχανισμό πιστοποίησης των αποτελεσμάτων μέσω των κρατουμένων «**εισόδου εις**» και «**εξόδου από**» το MSD.

6bit

Έλεγχος αποτελέσματος πρόσθεσης / αφαίρεσης μέσω κρατουμένων με την Παράσταση Συμπληρώματος ως προς 2

Ίδια κρατούμενα

π.χ. 16 + 2 = 18

-32	16	8	4	2	1	
0	1	0	0	0	0	= 16
+ 0	0	0	0	1	0	= 2
0	1	0	0	1	0	= **18**

0 0

Ίδια κρατούμενα

π.χ. -8+9= 1

-32	16	8	4	2	1	
1	1	1	0	0	0	= -8
+ 0	0	1	0	0	1	= 9
0	0	0	0	0	1	= 1

1 1

Overflow

π.χ. 22 + 22 = 44

-32	16	8	4	2	1	
0	1	0	1	1	0	= 22
+ 0	1	0	1	1	0	= 22
1	0	1	1	0	0	=-20≠44

0 1

Underflow

π.χ. -8-31=(-8)+(-31)=-39

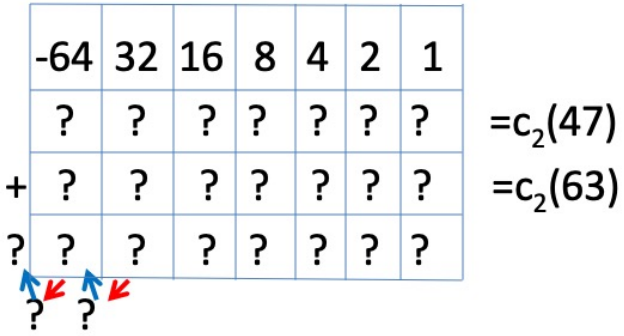
-32	16	8	4	2	1	
1	1	1	0	0	0	= -8
+ 1	0	0	0	0	1	= -31
0	1	1	0	0	1	=25≠-39

1 0

Παράδειγμα: Θέμα Εξετάσεων

- Να εκτελεστεί η πρόσθεση (αφού πρώτα μετατραπούν σε δυαδικούς) μεταξύ των αριθμών -47_{10} και -63_{10} σε δύο υπολογιστές Α & Β που χρησιμοποιούν την παράσταση συμπληρώματος ως προς 2. Ο Α έχει μήκος λέξης 7 bit ενώ ο Β 8 bit. Τι παρατηρείτε ?

1. Μετατροπή των 47_{10} και 63_{10} σε δυαδικ



2. Υπολογιστής Α: 7 bit

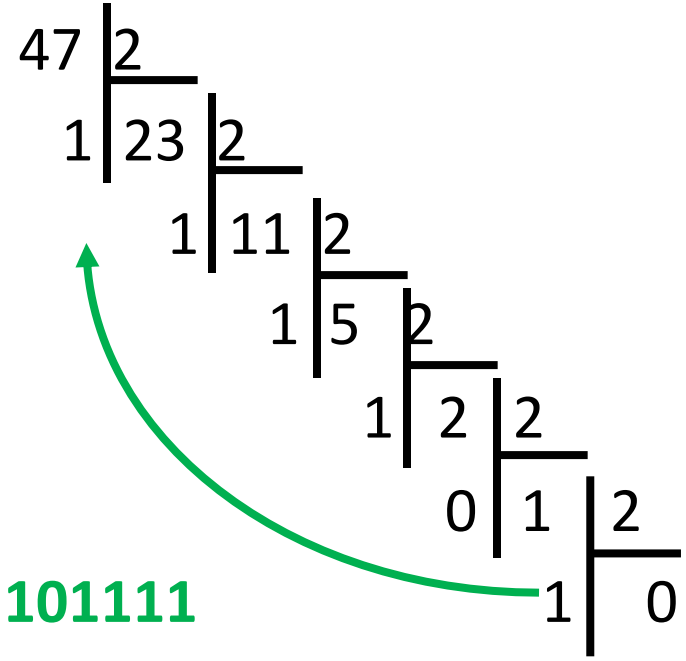
   2Α. Βρείτε το **c2(47)** και c2(**63**)

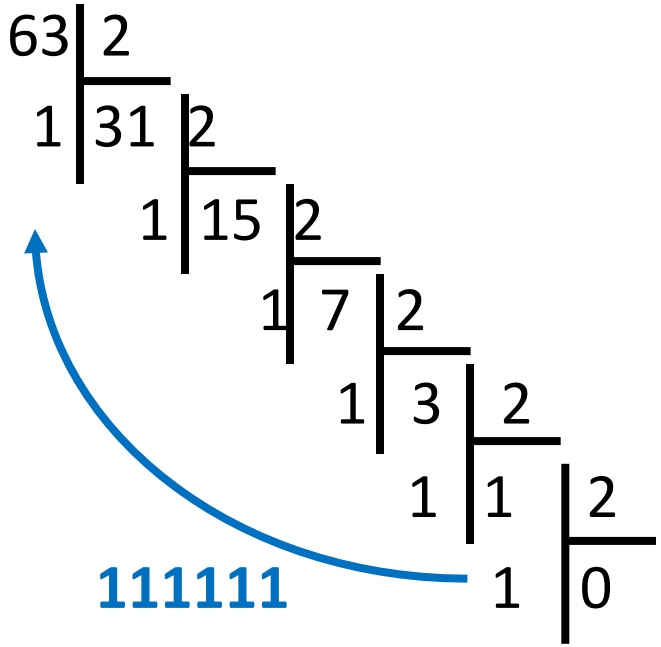
   2Β. Κάντε την πρόσθεση και ελέγξτε: Είναι το εισερχόμενο κρατούμενο στο MSD διαφορετικό από το εξερχόμενο?

3. Υπολογιστής Β: 8 bit – Επαναλάβετε την παραπάνω διαδικασία

Παράδειγμα: Θέμα Εξετάσεων

- Να εκτελεστεί η πρόσθεση (αφού πρώτα μετατραπούν σε δυαδικούς) μεταξύ των αριθμών **-47_{10}** και **-63_{10}** σε δύο υπολογιστές Α & Β που χρησιμοποιούν την παράσταση συμπληρώματος ως προς 2. Ο Α έχει μήκος λέξης 7 bit ενώ ο Β 8 bit. Τι παρατηρείτε ?
- Μετατροπή σε δυαδικό

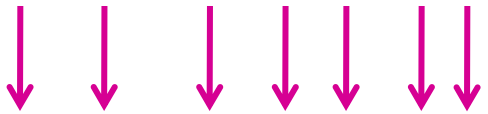




Παράδειγμα: συνεχ.

- $47_{10}=101111_2$ $63_{10}=111111_2$
- Πως παρίστανται τα -47_{10} ,-63_{10} ?
- Με συμπλήρωμα ως προς 2 αλλά με πόσα bit ?
- **Υπολογιστής Α**: 7 bit

6 5 4 3 2 1 0 : bit



-64	32	16	8	4	2	1	
0	1	0	1	1	1	1	$=47_{10}$
1	0	1	0	0	0	0	$=c_1(47)$
+						1	
1	0	1	0	0	0	1	$=c_2(47)$

-64	32	16	8	4	2	1	
0	1	1	1	1	1	1	$=63_{10}$
1	0	0	0	0	0	0	$=c_1(63)$
+						1	
1	0	0	0	0	0	1	$=c_2(63)$

Παράδειγμα: συνεχ.

-64 32 16 8 4 2 1

1 0 1 0 0 0 1 $=c_2(47)$

\+ 1 0 0 0 0 0 1 $=c_2(63)$

1 0 0 1 0 0 1 0 **Σωστό** ή **Λάθος** **???**

1 0 **ΛΑΘΟΣ**: γιατί το εισερχόμενο κρατούμενο στο MSD είναι διαφορετικό από το εξερχόμενο

- **Υπολογιστής Β**: 8 bit

-128 64 32 16 8 4 2 1

0 0 1 0 1 1 1 1 $=47_{10}$

1 1 0 1 0 0 0 0 $=c_1(47)$

\+ 1

1 1 0 1 0 0 0 1 $=c_2(47)$

-128 64 32 16 8 4 2 1

0 0 1 1 1 1 1 1 $=63_{10}$

1 1 0 0 0 0 0 0 $=c_1(63)$

\+ 1

1 1 0 0 0 0 0 1 $=c_2(63)$

Παράδειγμα: συνεχ.

-128 64 32 16 8 4 2 1

1 1 0 1 0 0 0 1 =c_2(47)

\+ 1 1 0 0 0 0 0 1 =c_2(63)

1 1 0 0 1 0 0 1 0 **Σωστό** ή **Λάθος**

1 1 **ΣΩΣΤΟ**: γιατί το εισερχόμενο κρατούμενο στο MSD είναι ίδιο με το εξερχόμενο

- **Επομένως**:
  - Επειδή -67+(-43)=-110 είναι λογικό να είναι
    - **λανθασμένη** η πράξη στον 7-μπιτο Η/Υ ο οποίος μπορεί να παραστήσει κατ' ελάχιστον τον αριθμό -64
    - **σωστή** η πράξη στον 8-μπιτο Η/Υ ο οποίος μπορεί να παραστήσει κατ' ελάχιστον τον αριθμό -128
  - Οι Η/Υ μέσω ελέγχου του κρατουμένου στο MSD έχουν τη δυνατότητα ελέγχου της ορθότητας των αποτελεσμάτων τους.

Τελος