ΗΥ-120: Ψηφιακή Σχεδίαση
Φθινόπωρο 2008

Τμ. Επ. Υπολογιστών

Εργαστήριο 6:
Προσημασμένοι Ακέραιοι, Προσθαφαιρέτες, Flip-Flops

3 - 7 Νοεμβρίου 2008
Διαγωνισμός Προόδου: Σάββατο 1 Νοεμβρίου 2008, 9:15 - 11:05 π.μ.

[Βιβλία: *προαιρετικά* μπορείτε να διαβάσετε: Wakerly: § **2.5 - 2.6** (σελ. 41-52)· Mano: § **1.5 - 1.6** (σελ. 13-24)].

6.1 Πολλαπλασιασμός, Διαίρεση, και Υπόλοιπο με δυνάμεις του 2

Όπως στο δεκαδικό σύστημα ο πολλαπλασιασμός επί 10, 100, κλπ. είναι πολύ απλός, έτσι και στο δυαδικό ο **πολλαπλασιασμός επί δύναμη του 2** αντιστοιχεί σε απλή **αριστερή ολίσθηση** με γέμισμα μηδενικών από δεξιά. Όντως, αν Β είναι ο δυαδικός αριθμός $b_{n-1}b_{n-2}...b_2b_1b_0$ με n bits (§5.1), τότε το γινόμενό του επί 2^k είναι:

$$B\cdot 2^k = b_{n-1}\cdot 2^{(n-1)+k} + b_{n-2}\cdot 2^{(n-2)+k} + ... + b_2\cdot 2^{2+k} + b_1\cdot 2^{1+k} + b_0\cdot 2^{0+k} =$$
$$= b_{n-1}\cdot 2^{n+k-1} + b_{n-2}\cdot 2^{n+k-2} + ... + b_1\cdot 2^{k+1} + b_0\cdot 2^k + 0\cdot 2^{k-1} + 0\cdot 2^{k-2} + ... + 0\cdot 2^1 + 0\cdot 2^0$$

Βάσει της τελευταίας αυτής ισότητας, ο αριθμός $B\cdot 2^k$ άναπαρίσταται στο δυαδικό με n+k bits τα οποία είναι: τα n bits του αριθμού Β "ολισθημένα" αριστερά κατά k θέσεις (δηλαδή με τη σημαντικότητα καθενός αυξημένη κατά k θέσεις), ακολουθούμενα από k μηδενικά. Γιά παράδειγμα, ο αριθμός 45 (δεκαδικό) = 101101 (δυαδικό), πολλαπλασιαζόμενος επί $16 = 2^4$ δίνει 45x16 = 720 (δεκαδικό), που στο δυαδικό είναι: 1011010000. Στο δεκαεξαδικό, ο πρώτος αριθμός (45) είναι ο 2D, και το γινόμενό του επί 16 (επί "10" στο δεκαεξαδικό) είναι: 2D0.

Κατ' αντίστοιχο τρόπο, η **διαίρεση διά δύναμη του 2** αντιστοιχεί σε **δεξιά ολίσθηση**· τα λιγότερο σημαντικά bits του αριθμού, που "εκδιώκονται" από τη δεξιά άκρη του, αποτελούν το **υπόλοιπο** της διαίρεσης. Γιά τον παραπάνω αριθμό Β, η διαίρεσή του διά 2^k δίνει:

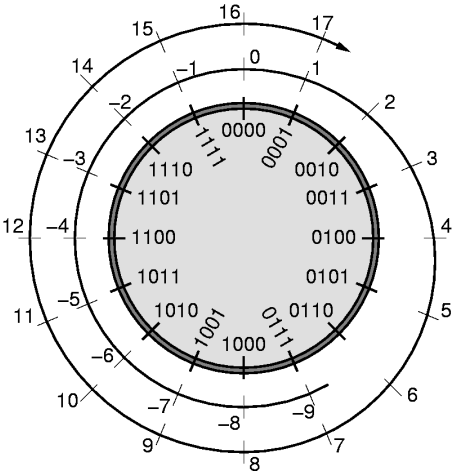
$$B / 2^k = (b_{n-1}\cdot 2^{(n-1)-k} + b_{n-2}\cdot 2^{(n-2)-k} + ... + b_{k+1}\cdot 2^1 + b_k\cdot 2^0) + (b_{k-1}\cdot 2^{k-1} + ... + b_1\cdot 2^1 + b_0\cdot 2^0) / 2^k$$

Ο πρώτος όρος του παραπάνω αθροίσματος είναι ακέραιος αριθμός, ενώ ο δεύτερος είναι κλασματικός, μικρότερος της μονάδας ή μηδέν, δεδομένου ότι ο αριθμητής $(b_{k-1}\cdot 2^{k-1} + ... + b_1\cdot 2^1 + b_0\cdot 2^0)$ είναι ένας δυαδικός αριθμός με k bits, άρα πάντα μικρότερος του 2^k. Κατά συνέπεια, ο πρώτος όρος (τα αριστερά *n-k* bits του αρχικού αριθμού) είναι το ακέραιο **πηλίκο** της διαίρεσης, ενώ ο αριθμητής του δευτέρου όρου (τα δεξιά *k* bits του αρχικού αριθμού) είναι το **υπόλοιπο** της (ακέραιας) διαίρεσης. Γιά παράδειγμα, ο αριθμός 205 (δεκαδικό) = 11001101 (δυαδικό), διαρούμενος διά $8 = 2^3$ δίνει πηλίκο 25 (δεκαδικό) = 11001 (δυαδικό), και υπόλοιπο 5 (δεκαδικό) = 101 (δυαδικό). Στο οκταδικό, ο διαιρετέος είναι 315, ο διαιρέτης είναι 10, το πηλίκο είναι 31 (=3x8+1=25 στο δεκαδικό), και το υπόλοιπο είναι 5.

Μιά συνηθισμένη εφαρμογή στους υπολογιστές είναι όταν μιά μεγάλη μνήμη, π.χ. 256 Mbytes, κατασκευάζεται από κάμποσες --π.χ. δεκαέξι (16)-- μικρότερες, μεγέθους δύναμης του 2 η καθεμία --εδώ μεγέθους 16 MBytes = 16,777,216 Bytes καθεμία. Εάν μας ζητηθεί να προσπελάσουμε το Byte με διεύθυνση 167,772,560, σε ποιάν από τις 16 μικρότερες μνήμες βρίσκεται αυτό και τι διεύθυνση μέσα σε αυτήν έχει; Γιά να βρούμε σε ποιά μνήμη θα το αναζητήσουμε, πρέπει να διαιρέσουμε τη διεύθυνση 167,772,560 διά το μέγεθος 16,777,216 των επιμέρους μνημών, που σ' αυτήν την περίπτωση μας δίνει 10 (δεκαδικό) (πρόκειται γιά την 11η μνήμη, αφού οι επιμέρους μνήμες αριθμούνται 0, 1, 2, ..., 15)· το υπόλοιπο της διαίρεσης, 400 (δεκαδικό), μας δίνει την επιθυμητή διεύθυνση μέσα στην επιμέρους μνήμη. Η διεύθυνση του επιθυμητού Byte είναι 167,772,560 (δεκαδικό) = A,00,01,90 (δεκαεξαδικό) = 1010,00000000,00000001,10010000 (δυαδικό - 28 bits). Το μέγεθος της κάθε επιμέρους μνήμης (διαιρέτης) είναι 16 M = 2^{24}, άρα το ζητούμενο Byte βρίσκεται στη θέση 00...0110010000 (υπόλοιπο διαίρεσης, δηλαδή δεξιά 24 bits της διεύθυνσης) της μνήμης υπ' αριθμόν 1010 (πηλίκο διαίρεσης, δηλαδή αριστερά 28-24 = 4 bits της διεύθυνσης)· μετατρέποντας στο δεκαδικό, βλέπουμε ότι όντως πρόκειται γιά τη θέση 256+128+16 = 400 της μνήμης υπ' αριθμόν 8+2 = 10.

6.2 Συστροφή (wrap-around) Αναπαράστασης Αριθμών με n bits

Μία άλλη εφαρμογή της παραπάνω παρατήρησης σχετικά με το υπόλοιπο της διαίρεσης διά δύναμη του 2 είναι η ερμηνεία των αριθμητικών πράξεων με πεπερασμένο πλήθος bits. Ας φανταστούμε ότι οι ακέραιοι αριθμοί των μαθηματικών έχουν πάρα πολλά bits --όσα χρειάζονται γιά το μέγεθός τους-- αλλά εμείς κάνουμε πράξεις μ' έναν υπολογιστή που έχει πεπερασμένο πλήθος bits, έστω n bits, και γι' αυτό κρατάμε μόνο τα n λιγότερο σημαντικά (δεξιά) bits του "μαθηματικού" αριθμού. Έτσι, όταν προσθέτουμε αριθμούς (§5.4) μ' έναν αθροιστή μεγέθους n bits, αγνοούμε (πετάμε) το κρατούμενο που βγαίνει από την αριστερή άκρη (MSB), διότι δεν έχουμε πού να το αποθηκεύσουμε. Επομένως, αν A είναι το πραγματικό (μαθηματικό) αποτέλεσμα της πράξης μας, ο υπολογιστής μας τελικά θα κρατήσει μόνο το **υπόλοιπο** της διαίρεσης του A διά 2^n (που συνήθως συμβολίζεται: A **mod** 2^n).

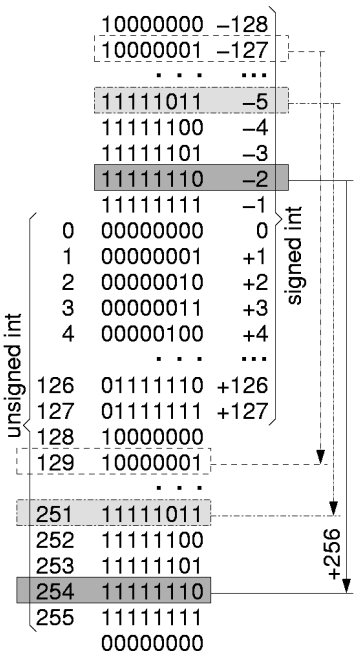


Εάν σε αυτό τον υπολογιστή, με το πεπερασμένο πλήθος των n bits, αρχίσω να μετρώ από το 0 προς τα πάνω, όταν φτάσω στον αριθμό 2^n-1, ο επόμενος αριθμός θα φανεί σαν να είναι πάλι ο 0, και μετά θα συνεχίσω να μετράω πάλι από την αρχή, 1, 10, 11, 100, κλπ. Το φαινόμενο αυτό είναι σαν να έχω ένα τροχό με περίμετρο 2^n και με χαραγμένους επάνω του τους 2^n υπάρχοντες συνδυασμούς των n bits, δεξιόστροφα από το 00...000 μέχρι το 11...111, και να **τυλίγω** επάνω σε αυτόν τον τροχό τον άξονα των αριθμών, όπως φαίνεται στο σχήμα δίπλα γιά n=4 bits. Έστω ότι ξεκινάμε το τύλιγμα έτσι ώστε ο ακέραιος αριθμός 0 να συμπέσει με τον δυαδικό κώδικα 00...000. Τότε, οι 2^n κώδικες του τροχού θα συμπέσουν με τους ακεραίους αριθμούς από 0 έως 2^n-1, όπως ακριβώς καθορίζει ο γνωστός μας από την §5.1 κώδικας δυαδικής αναπαράστασης των *μη προσημασμένων ακεραίων*. Όταν όμως εξαντλούνται οι υπάρχοντες συνδυασμοί των n bits, οι περαιτέρω αριθμοί (π.χ. 16, 17,... στο σχήμα) έχουν "τυλιχτεί" πάνω στους ίδιους κώδικες, 000, 001, κλπ, ξανά από την αρχή. Αν λοιπόν μιά πρόσθεση δώσει αποτέλεσμα από 2^n και πάνω, το αποτέλεσμα αυτό, στον υπολογιστή με τα n bits, θα μοιάζει σαν ένας μικρότερος αριθμός --αυτός που προκύπτει από το "τύλιγμα". Το φαινόμενο αυτό ονομάζεται "**wrap around**" --περιτύλιγμα, περιέλιξη, ή *συστροφή*-- των ακεραίων αριθμών γύρω από τον "τροχό" των πεπερασμένων συνδυασμών που έχει τη δυνατότητα να παραστήσει ο υπολογιστής.

Εάν τώρα θεωρήσουμε και τους αρνητικούς αριθμούς, στον ίδιο άξονα των αριθμών τον συνεστραμμένο γύρω από τον τροχό, όπως φαίνεται στο σχήμα, τότε αποκτάμε μιά μέθοδο --έναν κώδικα-- αναπαράστασης και αρνητικών αριθμών. Ο κώδικας αυτός, που ονομάζεται "συμπλήρωμα ως προς 2", χρησιμοποιείται σήμερα σε όλους τους υπολογιστές γιά την αναπαράσταση "προσημασμένων ακεραίων" (signed integers), και έχει πολύ σημαντικά πλεονεκτήματα απλότητας έναντι άλλων, εναλλακτικών κωδίκων. Στην καθημερινή μας ζωή παριστάνουμε τους προσημασμένους ακεραίους χρησιμοποιώντας έναν διαφορετικό κώδικα, τον κώδικα προσήμου - απόλυτης τιμής (sign-magnitude representation). Η κατεύθυνση αύξησης της απόλυτης τιμής, όμως, είναι άλλοτε δεξιόστροφα (θετικοί αριθμοί) και άλλοτε αριστερόστροφα (αρνητικοί αριθμοί), πάνω στον συνεστραμμένο άξονα των αριθμών. Αυτή η αλλαγή φοράς αύξησης έχει σαν συνέπεια, όταν προσθέτουμε προσημασμένους ακεραίους στην καθημερινή μας ζωή, άλλοτε να πρέπει να κάνουμε πρόσθεση κι άλλοτε αφαίρεση των απολύτων τιμών τους, και μάλιστα πριν από την αφαίρεση να πρέπει να συγκρίνουμε τις δύο απόλυτες τιμές γιά να βρούμε ποιά είναι η μικρότερη και να αφαιρέσουμε αυτήν από την άλλη.

Αντ' αυτού, οι υπολογιστές ακολουθούν τον πολύ απλούστερο τρόπο που πηγάζει από την παραπάνω μέθοδο της "συστροφής": επειδή η (αλγεβρική) αύξηση μιάς τιμής --είτε θετικής είτε αρνητικής-- αντιστοιχεί πάντα σε δεξιόστροφη κίνηση πάνω στον τροχό, προκύπτει ότι αρκεί πάντα να κάνουμε πρόσθεση και μόνο, ανεξαρτήτως του αν προσθέτουμε θετικούς ή αρνητικούς αριθμούς! Η άλλη βασική παρατήρηση είναι ότι η (αλγεβρική) ελάττωση μιάς τιμής, δηλαδή η πρόσθεση ενός αρνητικού αριθμού --π.χ. του (-1)-- που αντιστοιχεί σε αριστερόστροφη κίνηση πάνω στον τροχό --π.χ. κατά 22.5 μοίρες εδώ-- μπορεί να προκύψει ισοδύναμα και σαν πρόσθεση ενός "μεγάλου" θετικού αριθμού --του 15 στο εδώ παράδειγμα, που αντιστοιχεί σε δεξιόστροφη κίνηση κατά 360-22.5 = 337.5 μοίρες. Αν λοιπόν κωδικοποιήσουμε το -1 με τον ίδιο κώδικα όπως και το 15, τότε η πρόσθεση αυτού του κώδικα με έναν τετράμπιτο αθροιστή θα φέρνει το ίδιο αποτέλεσμα όπως η πρόσθεση του -1.

6.3 Κώδικας Συμπληρώματος-2 Προσημασμένων Ακεραίων

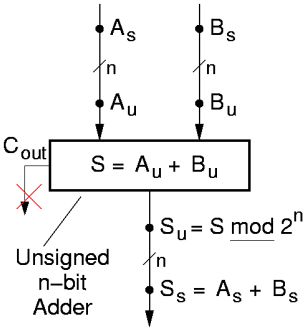


Σ' έναν υπολογιστή που λειτουργεί με λέξεις των n bits καθεμία, η συστροφή των ακεραίων αριθμών προκαλεί την επανεμφάνιση της ίδιας αναπαράστασης κάθε φορά που προχωρούμε κατά 2^n προς τα πάνω ή προς τα κάτω. Στο σχήμα δεξιά φαίνεται ένα παράδειγμα γιά έναν οκτάμπιτο υπολογιστή. Ο κώδικας 11111111, ερμηνευόμενος σαν μη προσημασμένος ακέραιος (unsigned integer), παριστά τον αριθμό 255 (δεκαδικό)· όμως, ο ίδιος κώδικας επανεμφανίζεται 256 θέσεις πιό πάνω, στο 255+256 = 511, όπως και 256 θέσεις πιό κάτω, στο 255-256 = -1. Η παρατήρηση αυτή είναι η βάση της δημιουργίας του κώδικα αναπαράστασης των προσημασμένων ακεραίων.

Ο κώδικας "**συμπληρώματος ως προς 2**" (2's Complement) που χρησιμοποιείται σήμερα στους υπολογιστές γιά την αναπαράσταση **προσημασμένων ακεραίων (signed integers)**, κωδικοποιεί τον αρνητικό αριθμό (-A), όπου A μεταξύ 1 και 2^{n-1}, με τον κώδικα μη προσημασμένου του ακεραίου **(-A)+2^n = 2^n-A**. Έτσι, στο σχήμα δεξιά, ο αριθμός -2 κωδικοποιείται όπως ο 2^n - 2 = 256 - 2 = 254, δηλαδή 11111110. Ομοίως, ο -3 κωδικοποιείται όπως ο 253, ο -4 όπως ο 252, ο -5 κωδικοποιείται 11111011, σαν τον 251, κ.ο.κ. Βλέπουμε ότι οι κώδικες των αρνητικών αριθμών αυξάνουν προς την ίδια κατεύθυνση προς την οποία αυξάνουν και οι κώδικες των θετικών αριθμών. Τους θετικούς αριθμούς από 0 έως 2^{n-1}-1, ο κώδικας συμπληρώματος ως προς 2 τους παριστά πανοποιότυπα όπως και ο κώδικας των μη προσημασμένων ακεραίων.

Σ' έναν οκτάμπιτο υπολογιστή όπως του παραπάνω παραδείγματος, έχουμε τη δυνατότητα να κωδικοποιήσουμε μονοσήμαντα μέχρι 256 διαφορετικούς ακεραίους αριθμούς· οι υπόλοιποι αριθμοί, που "δεν χωράνε" σε 8 bits, θα απεικονίζονται μέσω συστροφής σε κάποιον από τους "βασικούς" αριθμούς. Από τους άπειρους ακεραίους που υπάρχουν, ποιούς 256 θα διαλέξουμε σαν τους "βασικούς" ακεραίους, που θα χρησιμοποιούμε και θα κωδικοποιούμε μονοσήμαντα; Η απάντηση **εξαρτάται από τον κώδικα** που επιλέγουμε. Είδαμε στην §5.1 ότι ο οκτάμπιτος κώδικας μη προσημασμένων ακεραίων παριστάνει τους αριθμούς από το 0 ώς το 255· γενικότερα, με n bits, ο κώδικας αυτός παριστάνει τους ακεραίους από το 0 έως και το 2^n-1. Αντ' αυτού, ο κώδικας συμπληρώματος ως προς 2, με 8 bits, επιλέγουμε να παριστά μονοσήμαντα τους ακεραίους από -128 έως και +127· γενικότερα, με n bits, ο κώδικας αυτός παριστάνει τους ακεραίους από τον -2^{n-1} έως και τον $+2^{n-1}$-1. Τις περιοχές αυτές τις βλέπουμε σημειωμένες στο σχήμα με αγκύλες. Όπως βλέπουμε, η περιοχή αναπαράστασης του κώδικα των προσημασμένων ακεραίων είναι σχεδόν συμμετρική γύρω από το μηδέν, και έχει επιλεγεί ούτως ώστε το **αριστερό (MS) bit** του κώδικα να είναι **1** γιά όλους τους **αρνητικούς** αριθμούς και μόνο, και να είναι **0** γιά όλους τους **μη αρνητικούς** αριθμούς και μόνο, δηλαδή γιά τον αριθμό μηδέν και όλους τους θετικούς αριθμούς.

6.4 Πρόσθεση Προσημασμένων Ακεραίων σε Συμπλήρωμα-2

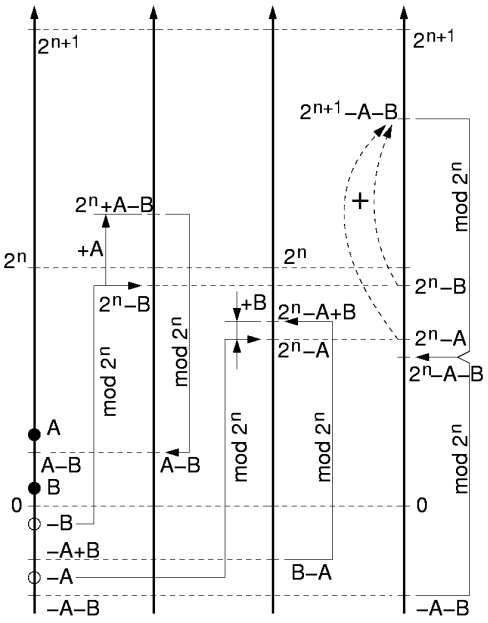


Θεωρήστε έναν υπολογιστή με λέξεις των n bits, και έναν αθροιστή μη προσημασμένων ακεραίων όπως αυτός της §5.4, του οποίου όμως αγνοούμε το κρατούμενο εξόδου, κρατάμε δηλαδή μόνο το άθροισμα **mod** 2^n, δηλαδή μόνο τα n LS bits. Θα αποδείξουμε ότι εάν στις δύο εισόδους αυτού του αθροιστή τροφοδοτήσουμε τους κώδικες συμπληρώματος-2 δύο προσημασμένων ακεραίων, A_S και B_S, μεταξύ -2^{n-1} και $+2^{n-1}$-1 ο καθένας, και εάν το (προσημασμένο) άθροισμα των δύο ακεραίων βρίσκεται επίσης στην περιοχή από -2^{n-1} έως και $+2^{n-1}$-1, τότε στην έξοδο του παραπάνω αθροιστή θα εμφανιστεί ο κώδικας συμπληρώματος-2 του (προσημασμένου) αυτού αθροίσματος. Με άλλα λόγια, ο αθροιστής *μη* προσημασμένων ακεραίων, όπως τον ξέρουμε, είναι επίσης και **αθροιστής προσημασμένων αριθμών**, όταν αυτοί παρίστανται με κώδικα συμπληρώματος-2, και όταν το άθροισμά τους χωρά να παρασταθεί και αυτό με τον ίδιο κώδικα και το ίδιο πλήθος bits.

Γιά την απόδειξη θα χρησιμοποιήσουμε, όπως φαίνεται στο παραπάνω σχήμα, τους ακεραίους A_u και B_u,

που αποτελούν την ερμηνεία των A_s και B_s ως μη προσημασμένων αριθμών· δηλαδή, όπως ξέρουμε, $A_u = A_s + 2^n$ όταν $A_s<0$, αλλοιώς $A_u = A_s$ --και ομοίως γιά τον B_u. Ο αθροιστής μη προσημασμένων υπολογίζει το $S_u = (A_u+B_u)$ **mod** 2^n, το οποίο στη συνέχεια ερμηνεύουμε σαν τον προσημασμένο αριθμό S_s, δηλ. $S_s = S_u$ όταν $S_u<2^{n-1}$, αλλοιώς $S_s = S_u - 2^n$. Η βασική ιδέα της απόδειξης είναι η εξής: αφού ο A_u είναι είτε A_s είτε $A_s + 2^n$, και ο B_u είναι είτε B_s είτε $B_s + 2^n$, τότε το άθροισμα (A_u+B_u) θα είναι: A_s+B_s+P, όπου P = είτε 0, είτε 2^n, είτε 2^{n+1}. Επομένως, αφού το άθροισμα S_u υπολογίζεται **mod** 2^n, ο όρος P θα φεύγει, και θα μας μένει το άθροισμα A_s+B_s. Γιά μιά πλήρη απόδειξη, πρέπει να εξετάσουμε προσεκτικά τις περιοχές τιμών των προσθετέων και του αθροίσματος, όπως θα κάνουμε τώρα, χωριστά γιά τις τέσσερεις περιπτώσεις:

(α) *Θετικός συν Θετικός:*
Όταν οι δύο προσθετέοι, A_s και B_s, είναι θετικοί ή μηδέν και δεν ξεπερνούν τον αριθμό $+2^{n-1}-1$, τότε $A_u = A_s$ και $B_u = B_s$, άρα και $A_u+B_u = A_s+B_s$. Η υπόθεση του θεωρήματός μας είναι ότι το άθροισμα A_s+B_s μπορεί να παρασταθεί με n bits σε μορφή συμπληρώματος 2, άρα το A_s+B_s δεν ξεπερνά το $+2^{n-1}-1$. Κατά συνέπεια, και το $S = A_u+B_u$ δεν ξεπερνά το $2^{n-1}-1$, επομένως $S_u = S <2^{n-1}$. Σε αυτήν την περιοχή, όμως, $S_s = S_u$, άρα $S_s = S = A_u+B_u = A_s+B_s$ [ΟΕΔ].

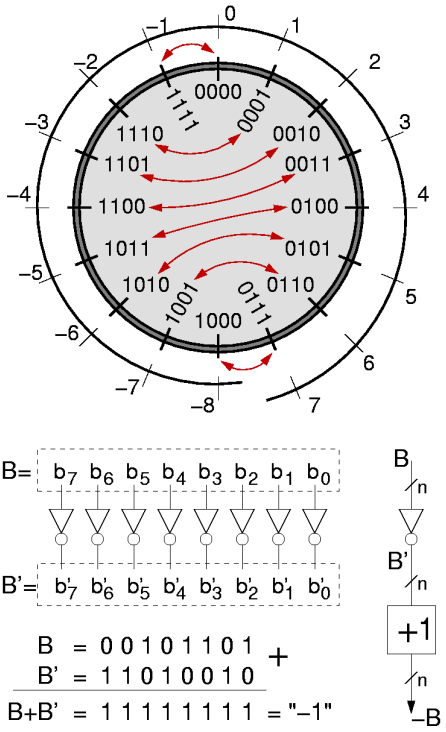


(β) *Θετικός συν Αρνητικό, με Άθροισμα Θετικό ή Μηδέν:*
Έστω ότι $A_s = +A$ και $B_s = -B$, όπου A και B είναι θετικοί που δεν ξεπερνούν το $+2^{n-1}-1$, και ο A είναι μεγαλύτερος ή ίσος του B, όπως φαίνεται στο σχήμα (πρώτοι δύο άξονες). Επειδή $A_s>0$, έχουμε $A_u=A_s=A$· απ' την άλλη μεριά, $B_s<0$, άρα $B_u = B_s+2^n = 2^n-B$. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = A + (2^n - B) = 2^n + (A-B)$, το οποίο όμως είναι μεγαλύτερο ή ίσο του 2^n, επειδή ο A είναι μεγαλύτερος ή ίσος του B. Άρα, $S_u = S$ **mod** $2^n = (2^n + (A-B))$ **mod** $2^n = A - B$. Επειδή $S_u=A-B$ δεν ξεπερνά το $+2^{n-1}-1$, θα είναι $S_s = S_u$· επομένως, $S_s = A-B = A_s+B_s$ [ΟΕΔ].

(γ) *Θετικός συν Αρνητικό, με Άθροισμα Αρνητικό:*
Έστω ότι $A_s = -A$ και $B_s = +B$, όπου A θετικός, B θετικός ή μηδέν, A>B, ο A δεν ξεπερνά το 2^{n-1}, και ο B δεν ξεπερνά το $2^{n-1}-1$. Σε αυτή την περίπτωση, έχουμε: $A_u = A_s+2^n = 2^n-A$, και $B_u=B_s=B$, όπως φαίνεται στο σχήμα, στη μετάβαση από τον πρώτο στον τρίτο άξονα των αριθμών. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = (2^n - A) + B = 2^n - (A-B)$· επειδή A>B, το άθροισμα αυτό είναι μικρότερο του 2^n, άρα ο αθροιστής το βγάζει αυτούσιο: $S_u = S = 2^n - (A-B)$. Επειδή ο A δεν ξεπερνά το 2^{n-1}, το ίδιο ισχύει και γιά τον (A-B)· επομένως, ο S_u είναι μεγαλύτερος ή ίσος του 2^{n-1}. Ένας τέτοιος μη προσημασμένος αριθμός, ερμηνευόμενος σε κωδικοποίηση συμπληρώματος-2, θα ερμηνευτεί σαν ο αρνητικός αριθμός $S_s = S_u - 2^n = (2^n-(A-B)) - 2^n = -A + B = A_s + B_s$ [ΟΕΔ].

(δ) *Αρνητκός συν Αρνητκό:*
Έστω ότι $A_s = -A$ και $B_s = -B$, όπου A και B είναι θετικοί που δεν ξεπερνούν, ούτε αυτοί ούτε το άθροισμά τους, τον αριθμό 2^{n-1}. Τότε: $A_u = A_s+2^n = 2^n-A$, και $B_u = B_s+2^n = 2^n-B$, όπως φαίνεται στο δεξιό άξονα του σχήματος. Ο μη προσημασμένος αθροιστής υπολογίζει το άθροισμα $S = (2^n-A) + (2^n-B) = 2^{n+1} - (A+B)$, το οποίο όμως είναι μεταξύ $2^{n+1}-2^{n-1} = 2^n+2^{n-1}$ και $2^{n+1}-1$. Άρα, επειδή ο αθροιστής βγάζει μόνο n bits, θα βγάλει τελικά: $S_u = S$ **mod** $2^n = (2^{n+1} - (A+B))$ **mod** $2^n = 2^n - (A+B)$, το οποίο είναι μεταξύ 2^{n-1} και 2^n-1. Σε αυτήν την περιοχή τιμών, $S_s = S_u - 2^n = (2^n - (A+B)) - 2^n = -A-B = A_s + B_s$ [ΟΕΔ].

6.5 Εύρεση του Αντιθέτου ενός Αριθμού



Διαπιστώσαμε ότι μπορούμε να προσθέτουμε προσημασμένους ακεραίους σε μορφή συμπληρώματος ως προς 2 χρησιμοποιώντας τον ίδιο αθροιστή μη προσημασμένων ακεραίων που ήδη είχαμε. Προκειμένου, τώρα, να κάνουμε και αφαιρέσεις, το μόνο που χρειαζόμαστε είναι μιά μέθοδος να βρίσκουμε τον αντίθετο αριθμό, -B, ενός δοθέντα αριθμού B. Όταν αποκτήσουμε μιά τέτοια μέθοδο, θα μπορούμε να κάνουμε την αφαίρεση A-B μέσω της πρόσθεσης A+(-B). Τη ζητούμενη μέθοδο μας τη δίνει η παρατήρηση του σχήματος: Έστω B ένας προσημασμένος ακέραιος σε μορφή συμπληρώματος-2 με n bits, και έστω B' ο επίσης προσημασμένος ακέραιος συμπληρώματος-2 με n bits που προκύπτει από τον B αντιστρέφοντας το κάθε bit του. Τον αριθμό B' τον λέμε και *"συμπλήρωμα του B ως προς 1" (1's complement)*, επειδή κάθε νέο bit είναι 1 μείον το παλαιό bit. Στο επάνω μέρος του σχήματος φαίνεται ο τροχός των προσημασμένων αριθμών της §6.2 με σημειωμένα επάνω τα ζευγάρια B - B' αριθμών που είναι ο ένας το συμπλήρωμα-ως-προς-1 του άλλου.

Εύκολα βλέπει κανείς ότι το προσημασμένο άθροισμα των B + B' ισούται πάντα με -1, αφού έχει την αναπαράσταση με όλο άσσους: η πρόσθεση ενός bit 0 με ένα bit 1, χωρίς κρατούμενο εισόδου, δίνει πάντα άθροισμα 1 και κρατούμενο 0 (χρησιμοποιήσαμε την ιδιότητα ότι οι προσημασμένοι αριθμοί B και B' προστίθενται με τον ίδιο τρόπο όπως προσθέτουμε και τους μη προσημασμένους αριθμούς, όπως αποδείξαμε στην προηγούμενη παράγραφο, 6.4). Αφού B+B' = -1, προκύπτει ότι B+B'+1 = B + (B'+1) = 0, άρα οι αριθμοί B και (B'+1) είναι αλγεβρικά αντίθετοι: (B'+1) = -B --υπό την προϋπόθεση ότι το άθροισμα (B'+1) μπορεί να παρασταθεί σε συμπλήρωμα-2 με n bits, δηλαδή ότι το (B'+1) βρίσκεται μεταξύ -2^{n-1} και $+2^{n-1}$-1, πράγμα που ισχύει πάντα εκτός της περίπτωσης B' = $+2^{n-1}$-1 = 0111...111, δηλαδή εκτός B = 1000...000 = -2^{n-1}, αφού τότε ο -B = $+2^{n-1}$ δεν μπορεί να παρασταθεί με n bits σε μορφή συμπληρώματος-2.

Επομένως, συνολικά, όταν **B** είναι ακέραιος στο διάστημα [-(2^{n-1}-1), +(2^{n-1}-1)], τότε ο αλγεβρικός αντίθετός του, **-B** (που επίσης ανήκει στο ίδιο διάστημα), σε παράσταση συμπληρώματος-2, είναι ο **B'+1**, δηλαδή ο αριθμός που προκύπτει από την παράσταση συμπληρώματος-2 του B αν αντιστρέψουμε το κάθε bit της και στη συνέχεια προσθέσουμε τον αριθμό 1 (μέσω προσημασμένης πρόσθεσης, που, όπως έχουμε δείξει, είναι η ίδια με την μη προσημασμένη όπου αγνοούμε το κρατούμενο εξόδου).

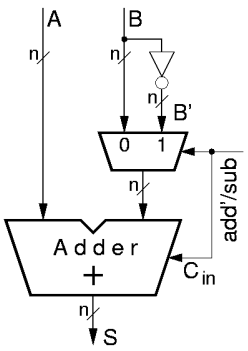
Άσκηση 6.6: Αρνητικοί Αριθμοί και Αφαιρέσεις

[Κάντε την πριν το εργαστήριο και παραδώστε την με την αναφορά σας.]
(α) Χρησιμοποιήστε τις παραστάσεις συμπληρώματος-2 του σχήματος της §6.3 γιά να κάνετε τις παρακάτω προσθέσεις μέσω του γνωστού αλγόριθμου **μη προσημασμένης πρόσθεσης**, με 8 bits και αγνοώντας το κρατούμενο εξόδου, και επιβεβαιώστε στο δεκαδικό ότι το αποτέλεσμα είναι σωστό: 32+(-1), 15+(-3), 2+(-5), (-1)+(-1), (-4)+(-1), (-127)+126, (-127)+127.
(β) Χρησιμοποιήστε τις παραστάσεις συμπληρώματος-2 του σχήματος της §6.3 και την παραπάνω μέθοδο εύρεσης του αντίθετου ενός αριθμού, γιά να βρείτε τα εξής αντίθετα, και επιβεβαιώστε την ορθότητα του αποτελέσματος: -(127), -(126), -(5), -(2), -(1), -(0) -(-1), -(-2), -(-3), -(-124), -(-126), και -(-127).

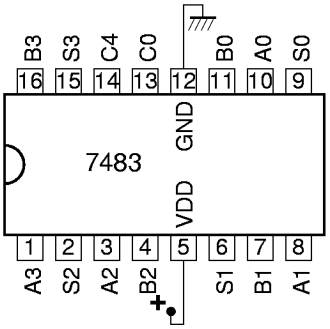
6.7 Προσθαφαιρέτες

Στους υπολογιστές συνήθως βρίσκει κανείς "προσθαφαιρέτες" (adder/subtracter) ακεραίων, κυκλώματα δηλαδή που μπορούν να κάνουν είτε πρόσθεση είτε αφαίρεση. Το κύκλωμα αυτό κατασκευάζεται πολύ εύκολα και με χαμηλό κόστος, με βάση έναν αθροιστή μη προσημασμένων ακεραίων (§5.4), όπως δείχνει το σχήμα. Ο αθροιστής φαίνεται στο κάτω μέρος του σχήματος: το σύμβολό του είναι ένα τραπέζιο με μιά εσοχή, έτσι που να θυμίζει ότι παίρνει δύο λέξεις δεδομένων, τις συνδυάζει, και βγάζει μία λέξη-απάντηση· μέσα στο τραπέζιο γράφεται το σύμβολο "+" γιά να ξεχωρίζουμε αυτόν τον αθροιστή

από άλλα κυκλώματα που κάνουν (και) άλλες πράξεις πάνω στις δύο εισόδους τους. Το κύκλωμα έχει δύο εισόδους δεδομένων, A και B, μία είσοδο ελέγχου, add'/sub, και μία έξοδο δεδομένων, S. Συνήθως τα δεδομένα είναι προσημασμένοι αριθμοί, αλλά μπορεί να χρησιμοποιηθεί και με μη προσημασμένους. Η είσοδος A δίδεται στον αθροιστή ως έχει· η είσοδος B, όμως, φτάνει στον αθροιστή με μία από δύο διαφορετικές μορφές, σύμφωνα με το τι επιλέγει ένας πολυπλέκτης 2-σε-1: είτε αυτούσια η λέξη B, είτε το συμπλήρωμά της ως προς 1, B'. Ο βασικός αθροιστής πρέπει να δέχεται κρατούμενο εισόδου, C_{in}. Όταν κάναμε πρόσθεση το κρατούμενο αυτό εισόδου ήταν πάντα 0, και γι' αυτό μπορούσαμε να το αγνοήσουμε χρησιμοποιώντας έναν ημιαθροιστή γιά τα δεξιά (LS) bits, όμως εδώ θα το χρειαστούμε αυτό το κρατούμενο εισόδου, (άρα χρειαζόμαστε πλήρεις αθροιστές στις θέσεις όλων των bits --και του LS). Γιά να κάνει το κύκλωμα πρόσθεση, S = A+B, θέτουμε την είσοδο ελέγχου add'/sub = 0· αυτό κάνει τη δεύτερη είσοδο του αθροιστή να ισούται με B, και το κρατούμενο εισόδου να είναι 0, όπως ακριβώς δηλαδή πρέπει γιά να γίνει η πρόσθεση A+B. Γιά να κάνει το κύκλωμα αφαίρεση, S = A-B, θέτουμε την είσοδο ελέγχου add'/sub = 1· αυτό κάνει τη δεύτερη είσοδο του αθροιστή να είναι το συμπλήρωμα B', και το κρατούμενο εισόδου να είναι 1. Δεδομένου ότι το κρατούμενο εισόδου έχει την ίδια σημαντικότητα, 2^0, με τα δεξιά (LS) bits των αριθμών εισόδου, A_0 και B_0, στα οποία και προστίθεται, το να ισούται το κρατούμενο εισόδου αυτό με 1 ισοδυναμεί με το να προστίθεται ο αριθμός 1 μαζί με τους δύο προσθετέους, A και B'. Επομένως, η έξοδος S = A + B' + 1· ξέρουμε όμως, από την §6.5, ότι B'+1 = -B, άρα η έξοδος S = A + (B'+1) = A + (-B) = A - B. Το όνομα του σήματος ελέγχου, add'/sub, είναι κατάλληλα γραμμένο ώστε να μας θυμίζει ότι όταν αυτό είναι αληθές (1) το κύκλωμα κάνει "sub" (αφαίρεση), ενώ όταν αυτό είναι ψευδές (0), δηλαδή add' ψευδές άρα add αληθές, τότε το κύκλωμα κάνει "add" (πρόσθεση).



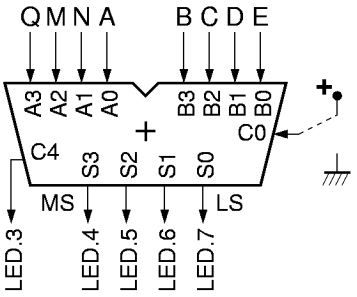
Πείραμα 6.8: Τετράμπιτος Αθροιστής



Caution: unconventional GND/VDD!

ΠΡΟΣΟΧΗ!: *το chip αυτού του πειράματος έχει τα pins τροφοδοσίας σε παράξενη, μη συμβατική θέση. Συνδέστε τα πολύ προσεκτικά, αλλοιώς θα κάψετε το chip!*

Στο πείραμα αυτό θα χρησιμοποιήσετε το chip "7483" το οποίο είναι ένας τετράμπιτος δυαδικός αθροιστής: περιέχει 4 πλήρεις αθροιστές (του ενός bit καθένας), με τα κρατούμενά τους συνδεδεμένα σε μιάν αλυσίδα. Λεπτομερείς πληροφορίες γιά το chip αυτό μπορείτε να βρείτε στη διεύθυνση που ανέφερε η §3.6. Τροφοδοτήστε τις εισόδους δεδομένων από τους 8 διακόπτες της πλακέτας εισόδων/εξόδων, και συνδέστε 5 LED's στις εξόδους, όπως στο σχήμα. Ελλείψει επαρκών διακοπτών εισόδου, συνδέστε την είσοδο κρατουμένου του αθροιστή σε μία από τις τάσεις τροφοδοσίας με ένα σύρμα που μπορείτε να το εναλλάσετε χειροκίνητα μεταξύ γής (0) και θετικού (1). **Πριν** φτάσετε στο εργαστήριο, συμπληρώστε τον παρακάτω πίνακα:

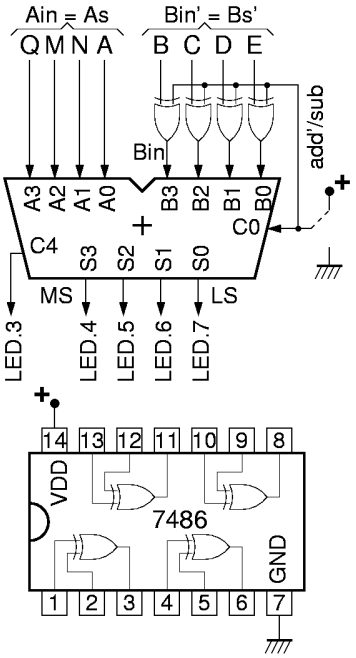


Ain	Bin	Cin	Au+Bu+Cin	S(5b)	As+Bs+Cin	S(4)	Bin' =Bs'	As-Bs'=S(4)
0010	0011	1	2+3+1 = 6	00110	2+3+1 = 6	0110	1100 = -4	2-(-4) = 6
0010	1011	1	2+11+1=14	01110	2-5+1 =-2	1110	0100 = +4	2-(+4) =-2
1110	1111	1						

{{επόμενες γραμμές}}:

όπου οι {{επόμενες γραμμές}} είναι: 1111 + 1111, 0101 + 1110, 0101 + 0001, 0101 + 0000, και όλες με Cin=1. Στον πίνακα αυτόν, Ain, Bin, και Cin είναι οι δυαδικές είσοδοι του αθροιστή· Au+Bu+Cin είναι η ερμηνεία των εισόδων και της αναμενόμενης εξόδου σύμφωνα με τον κώδικα μη προσημασμένων αριθμών· S(5b) είναι η αναμενόμενη πεντάμπιτη έξοδος του αθροιστή (άθροισμα μη προσημασμένων εισόδων), και πρέπει να συμφωνεί με την προηγούμενη στήλη· As+Bs+Cin είναι η ερμηνεία των εισόδων και της αναμενόμενης εξόδου σύμφωνα με τον κώδικα συμπληρώματος-2 προσημασμένων αριθμών· S(4) είναι η αναμενόμενη τετράμπιτη έξοδος του αθροιστή (άθροισμα προσημασμένων εισόδων), και πρέπει να συμφωνεί με την προηγούμενη στήλη· Bin' είναι το συμπλήρωμα ως προς 1 της εισόδου Bin --ας θεωρήσουμε από 'δω και πέρα ότι αυτό ήταν η αρχική είσοδος ενός αφαιρέτη, πριν ένας αντιστροφέας δώσει το Bin στον αθροιστή, και Bs' είναι η ερμηνεία αυτού του Bin' σαν προσημασμένου αριθμού· As-Bs'=S(4) είναι η ερμηνεία της πράξης του (φανταστικού) αφαιρέτη, και πρέπει να συμφωνεί με τη στήλη S(4). **Στο εργαστήριο**, επαληθεύστε πειραματικά τις εξόδους S(5b) και S(4) του πίνακα. **Μην** χαλάστε το κύκλωμά σας όταν τελειώσετε: θα το χρειαστείτε στο πείραμα 6.9.

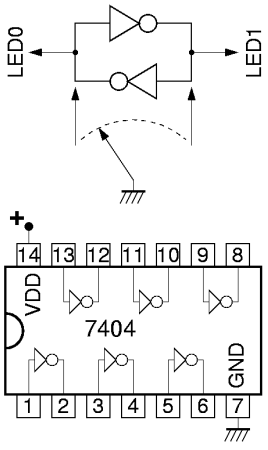
Πείραμα 6.9: Προσθαφαιρέτης με XOR



Ο προσθαφαιρέτης της §6.7 μπορεί εναλλακτικά να υλοποιηθεί με πύλες αποκλειστικού-Ή (XOR), όπως φαίνεται στο σχήμα δίπλα, αντί αντιστροφέων και πολυπλέκτη. Ο λόγος είναι ότι οι πύλες XOR δίνουν στην έξοδό τους την πρώτη είσοδο όταν η δεύτερη είσοδος είναι 0, ενώ δίνουν στην έξοδό τους το συμπλήρωμα της πρώτης εισόδου τους όταν η δεύτερη είσοδος είναι 1. Φτιάξτε έναν τετράμπιτο προσθαφαιρέτη, χρησιμοποιώντας τον αθροιστή του προηγουμένου πειράματος 6.8 και ένα chip πυλών XOR (7486) όπως αυτό του πειράματος 5.7· γιά διευκόλυνσή σας, η διάταξη ακροδεκτών του επαναλαμβάνεται στο σχήμα εδώ. **Πριν** φτάσετε στο εργαστήριο, κάντε το σχεδιάγραμμα συνδεσμολογίας που δείχνει ποιά συγκεκριμένα pins ποιού chip πρέπει να συνδέσετε πού. **Στο εργαστήριο**, κατασκευάστε τον προσθαφαιρέτη, και δώστε του τις εισόδους Bin' του πίνακα του παραπάνω πειράματος 6.2, με το σήμα add'/sub = 1, ούτως ώστε να διαπιστώσετε πως αυτές οι αφαιρέσεις γίνονται σωστά. Μετά, γυρίστε το σήμα add'/sub στο 0 (πρόσθεση), και κάντε κάμποσες προσθέσεις γιά να διαπιστώσετε τη σωστή λειτουργία --π.χ. κάντε τις προσθέσεις των δύο πρώτων στηλών του παραπάνω πίνακα, χωρίς κρατούμενο εισόδου αυτή τη φορά.

Πείραμα 6.10: Η στοιχειώδης Ιδέα του Flip-Flop

Στο πείραμα 2.10 είχαμε δεί ότι τα κυκλώματα με θετική ανάδραση έχουν συνήθως δύο σταθερές καταστάσεις, κι έτσι χρησιμοποιούνται σαν μνήμες. Το απλούστερο κύκλωμα λογικών πυλών που έχει θετική ανάδραση είναι δύο αντιστροφείς (πύλες NOT) συνδεδεμένοι κυκλικά ώστε να τροφοδοτούν ο ένας τον άλλον, όπως στο σχήμα. Η ανάδραση είναι θετική επειδή δύο αρνήσεις κάνουν μία κατάφαση. Προφανώς, το κύκλωμα αυτό έχει δύο σταθερές καταστάσεις: *(i)* ο αριστερός κόμβος μπορεί να είναι 0 και ο δεξιός 1, ή *(ii)* ο αριστερός κόμβος μπορεί να είναι 1 και ο δεξιός 0. Το κύκλωμα αυτό αποτελεί τη βασική ιδέα του *flip-flop*, δηλαδή του στοιχειώδους κυττάρου ψηφιακής μνήμης.



Κατασκευάστε το κύκλωμα αυτό στο εργαστήριο, και συνδέστε δύο LED's στις εξόδους του. Χρησιμοποιήστε δύο από τις πύλες ενός chip 7404 (βλ. §3.6). Το κύκλωμα αυτό δεν έχει εξωτερικές εισόδους, αφού κάθε είσοδος πύλης του οδηγείται από μιάν άλλη πύλη του! Επομένως, δεν μπορείτε να του αλλάξετε την κατάστασή του. Σβήστε και ανάψτε την τροφοδοσία πολλές φορές, και παρατηρήστε αν το κύκλωμα "σηκώνεται" πάντα στην ίδια κατάσταση, ή πότε στη μιά και πότε στην άλλη. (Αν το κύκλωμα είναι εντελώς "συμμετρικό", από ηλεκτρική άποψη, θα "πέφτει" με πιθανότητα 50% στη μία κατάσταση και 50% στην άλλη, σαν μιά μπίλια που την τοποθετείτε ακριβώς ισορροπημένη στην κόψη ενός ξυραφιού, κι αυτή πέφτει πότε από τη μιά και πότε από την άλλη· αν όμως το κύκλωμα έχει έστω και μία ανεπαίσθητη (ηλεκτρική) ασυμμετρία, τότε θα πέφτει συνήθως από την ίδια "πλευρά").

Ένας --ημιπαράνομος...-- τρόπος να επηρρεάσουμε την κατάσταση του flip-flop (με την τροφοδοσία αναμένη) είναι να ακουμπήσουμε "στιγμιαία" τον έναν από τους δύο κόμβους του με ένα σύρμα γειωμένο, όπως δείχνει το σχήμα με τα κατακόρυφα βέλη. (Εναλλακτικά, το σύρμα θα μπορούσε να συνδέονταν και στην θετική τροφοδοσία). (Γενικά, σ' ένα τυχόν κύκλωμα, είναι πολύ κακό να ακουμπάς την έξοδο μιάς πύλης στη γή ή στην τροφοδοσία, διότι όταν η έξοδος προσπαθεί να δώσει την αντίθετη τιμή από αυτήν που της ακουμπάμε, κατ' ουσίαν προκαλούμε βραχυκύκλωμα και υπερθέρμανση, με πιθανό κάψιμο αν συνεχιστεί γιά πολλήν ώρα· εδώ, ευτυχώς, τα πράγματα δεν είναι τόσο κακά: το βραχυκύκλωμα διαρκεί ελάχιστα ns μόνο, μέχρις ότου η νέα τιμή που αυτό επιβάλει εξωτερικά κάνει την "βόλτα" του κυκλώματος και αλλάξει κατά την ίδια φορά την τιμή που η πύλη προσπαθεί να δώσει στην έξοδό της· έτσι, η υπερθέρμανση είναι πολύ μικρή). Ακουμπήστε τον έναν κόμβο, και δείτε ότι αμέσως η αντίστοιχη LED σβήνει και η άλλη ανάβει. Απομακρύνετε το σύρμα και παρατηρήστε ότι το κύκλωμα μένει εκεί που το αφήσατε, δηλαδή έχει **μνήμη**. Επαναλάβετε από την άλλη πλευρά. Όταν το ακούμπημα είναι από την ίδια πλευρά που είναι ήδη σβηστή, αυτό δεν έχει επίδραση· όταν είναι από την άλλη, τότε έχει. Αυτή είναι η βασική (χειροκίνητη) ιδέα του flip-flop τύπου RS, την αυτοματοποίηση της οποίας (με πύλες αντί ακουμπήματος συρμάτων) θα δούμε στο επόμενο εργαστήριο.